

Gestione Clienti Bancari (Lista Concatenata 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 record relativi a clienti bancari. Ciascun cliente è caratterizzato da nome, cognome, numero di conto, anno di apertura e saldo del conto.

Il programma deve ricevere i file di input e output tramite riga di comando, caricare i dati in una lista ordinata per anno di apertura, calcolare una specifica metrica finanziaria per rimuovere ciclicamente i clienti con punteggio più elevato, travasare i clienti restanti in un array dinamico, ordinarli per saldo decrescente e salvare il risultato in un file di testo.

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. `input.txt`) e il nome del file di output (es. `output.txt`).
- Entrambi i file devono avere estensione `.txt`. 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 Lista Concatenata:

- **Punto 2a [4 punti]:** Definire la struttura `Account` (nome e cognome come stringhe max 255 caratteri, numero di conto

come intero, anno di apertura conto come intero, saldo come float) e il nodo **Node** per gestire una lista concatenata semplice.

- **Punto 2b [4 punti]:** Aprire il file di input, leggerne le righe ed effettuarne il parsing (campi separati da spazi), popolando la lista in modo che rimanga ordinata per anno di apertura del conto in ordine crescente. Se il file non può essere aperto, stampare un errore ed uscire con exit code 3. Stampare a schermo la lista inserita.

PUNTO C: [8 punti] Calcolo Metrica e Rimozione Ciclica:

- **Punto 3a [4 punti]:** Determinare il saldo massimo m contenuto nella lista. Calcolare per ciascun cliente il coefficiente x definito come:

$$x = \min \left(\frac{2023 - \text{anno_apertura}}{5}, 1 \right) \cdot \frac{\text{saldo}}{m}$$

Identificare il cliente con il più alto valore di x .

- **Punto 3b [4 punti]:** Rimuovere il cliente con il valore di x massimo dalla lista. Questa operazione deve essere ripetuta h volte, con $h = \lfloor n/2 \rfloor$ (parte intera inferiore di $n/2$), dove n è la lunghezza iniziale della lista. Stampare le informazioni di ciascun cliente rimosso durante i cicli.

PUNTO D: [8 punti] Travaso su Array ed Ordinamento:

- **Punto 4a [4 punti]:** Svuotare la lista dinamica degli elementi rimasti, trasferendoli in un array di clienti allocato dinamicamente con `malloc` della dimensione esatta.
- **Punto 4b [4 punti]:** Ordinare l'array in-place in base al saldo in ordine decrescente utilizzando l'algoritmo **Insertion Sort** (o un altro algoritmo in-place). Stampare a schermo l'array ordinato.

PUNTO E: [3 punti] Output Formattato su File: Scrivere i clienti rimasti e ordinati nel file di output.

- Scrivere nel file ciascun record rispettando lo stesso formato del file di input (campi separati da spazio).
- 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 con opportuni parametri formali:

- **decodeParameters**: riceve `argc` e `argv`, valida i parametri e li restituisce tramite puntatori o struttura dedicata.
- **readFile**: legge il file di input e popola la lista concatenata ordinatamente, restituendone la testa.
- **getMaxBalance**: calcola e restituisce il saldo massimo contenuto nella lista concatenata.
- **removeBestAccount**: calcola x per ciascun elemento, identifica l'elemento con x massimo, stampa i suoi dettagli e lo rimuove dalla lista.
- **listToArray**: svuota la lista copiandone gli elementi in un array allocato dinamicamente, restituendone il puntatore e aggiornandone la dimensione.
- **insertionSort**: ordina l'array di clienti in base al saldo in ordine decrescente.
- **writeOutput**: scrive il contenuto dell'array sul file di output.

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 input.txt output.txt
```

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

```
===== Lista Iniziale (Ordinata per Anno Apertura) =====
Amelia Kearns 98277620 1998 982.99
Jennifer Kim 1726355 1998 1293.88
Alicia Kiser 982736 2001 145.45
Eleanor Kerns 827563900 2002 8273.28
Abigail Kelley 298367 2008 8273.15
Amber King 7263529 2012 10283.22
Amy Kelly 92837102 2015 87654.11
Heather Knutson 2983620 2019 2736.86
Alexandra Kennedy 89201736 2021 9287.33
Anna Knight 8827263 2022 8973.76

===== Rimozione dei 5 Clienti con X massimo =====
Rimosso account: Amy Kelly 92837102 2015 87654.11 (x = 1.0000)
Rimosso account: Amber King 7263529 2012 10283.22 (x = 1.0000)
Rimosso account: Alexandra Kennedy 89201736 2021 9287.33 (x = 0.3612)
Rimosso account: Eleanor Kerns 827563900 2002 8273.28 (x = 0.9441)
Rimosso account: Abigail Kelley 298367 2008 8273.15 (x = 0.9441)

===== Array Rimanente Ordinato per Saldo Decrescente =====
Anna Knight 8827263 2022 8973.76
Heather Knutson 2983620 2019 2736.86
Jennifer Kim 1726355 1998 1293.88
Amelia Kearns 98277620 1998 982.99
Alicia Kiser 982736 2001 145.45

Output salvato con successo in output.txt
```

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

```
Anna Knight 8827263 2022 8973.76
Heather Knutson 2983620 2019 2736.86
Jennifer Kim 1726355 1998 1293.88
Amelia Kearns 98277620 1998 982.99
Alicia Kiser 982736 2001 145.45
```

(Exit code atteso: 0)

Caso 2 — Esecuzione con estensione parametri errata

Eseguendo il programma con il comando:

```
./main input.txt output.pdf
```

Il programma stampa su `stderr`:

```
Errore: i nomi dei file devono avere estensione ".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 lista con dati statici e proseguire. L'uso di questa funzione azzerà il punteggio del Punto 2b.

```
void carica_dati_mock(Node** head) {
    Account test_data[] = {
        {"Amber", "King", 7263529, 2012, 10283.22},
        {"Anna", "Knight", 8827263, 2022, 8973.76},
        {"Amy", "Kelly", 92837102, 2015, 87654.11},
        {"Alexandra", "Kennedy", 89201736, 2021, 9287.33},
        {"Abigail", "Kelley", 298367, 2008, 8273.15}
    };
    for(int i = 0; i < 5; i++) {
        insertOrdered(head, test_data[i]);
    }
    printf("\n[!] Dati di emergenza caricati (Bypass Lettura File).\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:

```
Account 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.