

Laboratorio 6 — Gestione di un Piano di Studi Binario

Live Demo — studyplan.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 `studyplan.c`, che gestisca un archivio binario delle materie di un piano di studi universitario. Ogni materia è descritta da quattro campi memorizzati in forma interlacciata all'interno del file binario:

- il nome della materia, come stringa a lunghezza massima prefissata;
- il numero di crediti, come intero;
- il voto, come intero;
- il flag di lode, come singolo carattere.

Il file binario non contiene strutture C dichiarate tramite `struct`, ma una sequenza di record a formato fisso scritti in ordine noto. Ogni record deve essere memorizzato sul file nel seguente ordine sequenziale:

1. nome della materia, come blocco di `MAX_NAME` (64) byte;
2. crediti, come `int` (4 byte);
3. voto, come `int` (4 byte);
4. lode, come `char` (1 byte).

La dimensione fisica di ciascun record memorizzato sul disco rigido è pertanto pari a esattamente $64 + 4 + 4 + 1 = 73$ byte.

Si assumano i seguenti vincoli dimensionali:

```
#define MAX_NAME 64
#define MAX_SUBJ 256
```

Il programma può operare in memoria tramite quattro array paralleli:

```
char names[MAX_SUBJ][MAX_NAME];
int credits[MAX_SUBJ];
int grades[MAX_SUBJ];
char honors[MAX_SUBJ];
```

Requisiti di Implementazione

Il programma deve essere strutturato nei seguenti punti:

PUNTO A: Argomenti da riga di comando: Verificare che siano passati esattamente due argomenti oltre all'eseguibile: la modalità operativa (**new**, **append**, **print**, **sort**, **stats**) e il nome del file binario. In caso di errore nel numero di argomenti, stampare su **stdout**:

```
Errore: uso corretto ./studyplan [new|append|print|sort|stats]
      ↪ file.bin
```

e terminare con exit status 1. Se la modalità non è tra le 5 previste, stampare:

```
Errore: modalita non riconosciuta.
```

e terminare con exit status 1.

PUNTO B: Validazione CSV: Implementare una funzione di supporto per il parsing e la validazione di una riga nel formato CSV:

```
NomeMateria,Crediti,Voto,Lode
```

Usare **strtok** con separatore **,.** Regole di validazione:

- Nome materia: lunghezza compresa tra 1 e **MAX_NAME - 1** caratteri.
- Crediti: intero strettamente positivo (> 0).
- Voto: intero compreso tra 18 e 30 inclusi.
- Lode: carattere 'L' oppure vuoto (in tal caso memorizzare lo spazio ' ').

In caso di riga malformata o dati non validi, stampare:

```
Errore: riga CSV non valida.
```

e terminare con exit status 1.

PUNTO C: Modalità new: Aprire il file binario in modalità scrittura ("**wb**"), leggere da **stdin** una sequenza di righe CSV fino a EOF (usando **fgets**), validare e popolare gli array paralleli. Scrivere infine l'intero archivio record dopo record. Se le materie superano **MAX_SUBJ**, stampare:

Errore: numero massimo di materie superato.

e uscire con 1. In caso di successo, stampare **Archivio creato con successo.** e uscire con 0.

PUNTO D: Modalità append: Aprire il file in modalità append binaria ("ab"). Leggere una sola riga da **stdin**, validarla, e scriverla in coda al file. In caso di errore di apertura, stampare:

Errore: impossibile aprire il file binario.

e uscire con 1. In caso di successo, stampare **Materia aggiunta con successo.** e uscire con 0.

PUNTO E: Modalità print: Aprire in lettura binaria ("rb"), determinare la dimensione del file con **fseek** e **ftell**. Verificare che sia un multiplo esatto di 73 byte. Stampare le materie nel formato:

Elenco materie:
Nome: [Nome] | Crediti: [C] | Voto: [Voto][Lode]

Se il file è vuoto, stampare:

Elenco materie:
Nessuna materia presente.

e terminare con exit status 0.

PUNTO F: Modalità sort: Leggere l'intero archivio dal file binario in memoria negli array paralleli, ordinarli in ordine crescente di voto (in caso di parità, ordinamento lessicografico del nome) e riscrivere il file ordinato in modalità "wb". Al termine, stampare **Archivio ordinato con successo.** e uscire con 0.

PUNTO G: Modalità stats: Leggere i record dal file binario, calcolare e stampare: numero di materie, crediti totali, media pesata arrotondata alla seconda cifra decimale, materia migliore e peggiore (con voto e lode se presente), e numero di lodi. Formula della media pesata:

$$\text{media pesata} = \frac{\sum (\text{voto}_i \cdot \text{crediti}_i)}{\sum \text{crediti}_i}$$

Se l'archivio è vuoto, stampare **Archivio vuoto.** ed uscire con 0.

Output di Controllo (Esempi di Esecuzione)

Caso 1 — Creazione e visualizzazione valida dell'archivio

Invocando:

```
./studyplan new piano.bin
```

ed inserendo da standard input:

```
Programmazione 1,9,30,L  
Analisi Matematica,12,27,  
Fisica,6,24,
```

seguito da EOF, il programma deve stampare a schermo:

```
Archivio creato con successo.
```

(Exit code atteso: 0)

Invocando poi:

```
./studyplan print piano.bin
```

il programma deve stampare a schermo:

```
Elenco materie:  
Nome: Programmazione 1 | Crediti: 9 | Voto: 30L  
Nome: Analisi Matematica | Crediti: 12 | Voto: 27  
Nome: Fisica | Crediti: 6 | Voto: 24
```

(Exit code atteso: 0)

Caso 2 — Calcolo statistiche

Invocando:

```
./studyplan stats piano.bin
```

il programma deve stampare a schermo:

```
Numero materie: 3  
Crediti totali: 27  
Media pesata: 27.33  
Materia migliore: Programmazione 1 (30L)  
Materia peggiore: Fisica (24)  
Materie con lode: 1
```

(Exit code atteso: 0)

Caso 3 — Inserimento riga non valida

Invocando:

```
./studyplan new piano.bin
```

ed inserendo da standard input:

```
Programmazione 1,9,trenta,L
```

il programma deve stampare a schermo:

```
Errore: riga CSV non valida.
```

(Exit code atteso: 1)