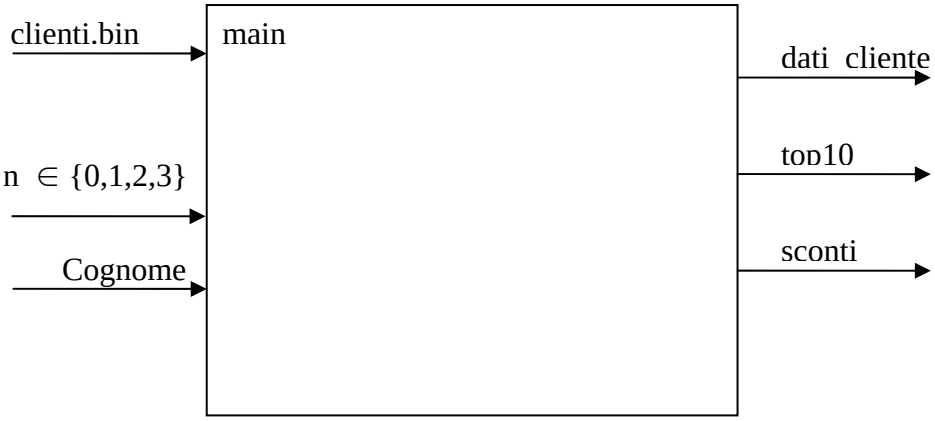
Fondamenti di Informatica L-A (Elettronica/Automazione)
A.A. 2005/2006, terzo scritto (18/1/2005)
Proposta di soluzione (ragionata) dell'esercizio di progetto

Input/Output
Input = clienti.bin, n (funzione richiesta dall'utente), Cognome (per il punto 1)
Output = dati_cliente, top10, sconti



Funzionalità richieste

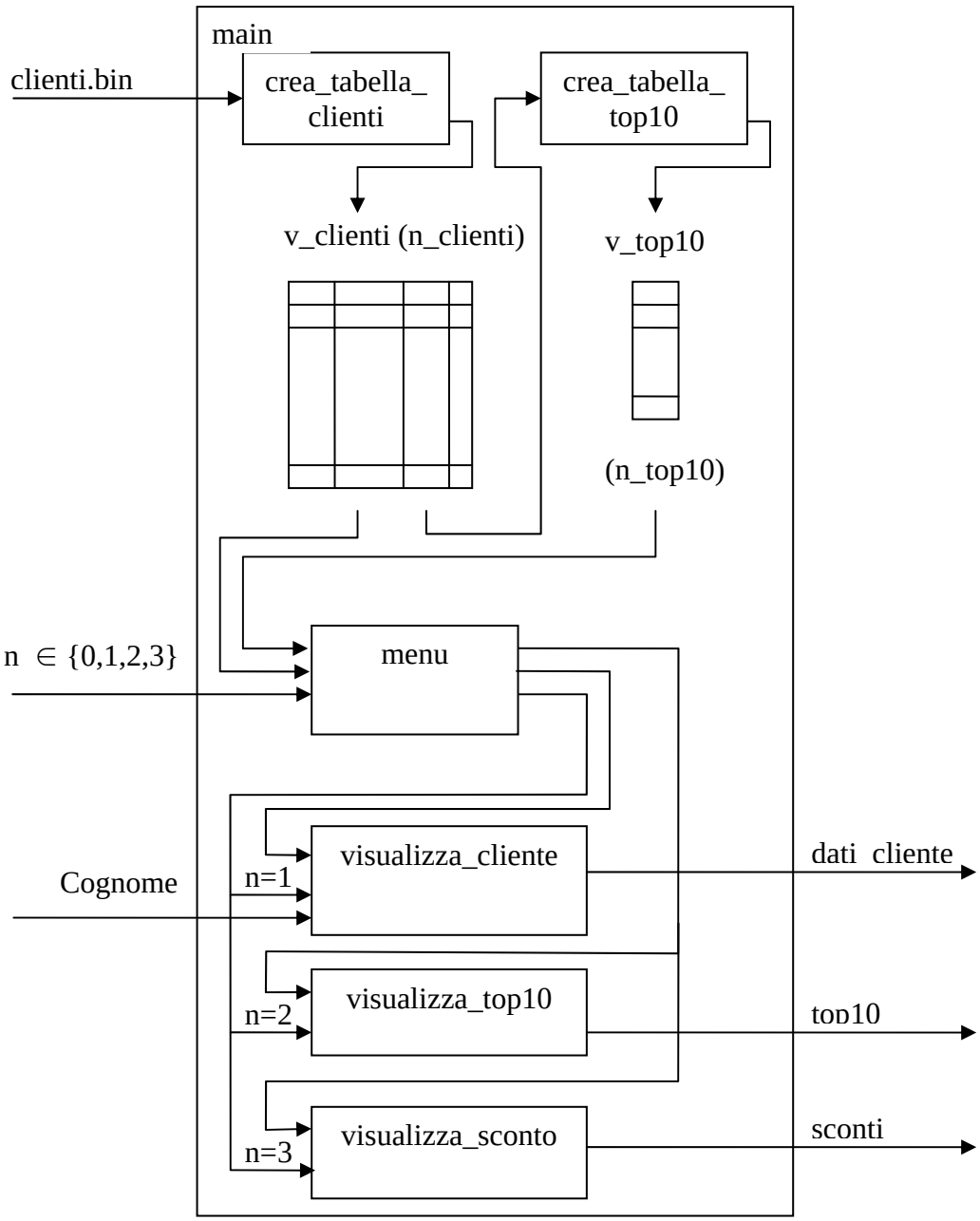
1. Visualizzazione di ID e spesa (CD e DVD) di un cliente. Mi servono:
   - il cognome del cliente
   - la somma della spesa del cliente in CD
   - la somma della spesa del cliente in DVD
2. Visualizzazione dell'elenco dei clienti più fedeli ("top10"). Mi servono:
   - un elenco dove posso conservare 10 riferimenti a clienti
   - le spese di ciascun cliente (somma di CD e DVD in promozione)
3. Visualizzazione dello sconto. Mi servono:
   - le spese di ciascun cliente sugli articoli in promozione
   - l'elenco dei top10 (punto 2)

Considerazioni iniziali di progetto

- I vari punti richiedono di svolgere operazioni sui dati contenuti nel file clienti.bin. In particolare:
  - ⇨ siccome è richiesto un menu, devo considerare che le varie operazioni potranno essere eseguite più volte
  - ⇨ nel punto 2: per determinare i "top10" dovrò selezionare alcuni tra i clienti
  - ⇨ nel punto 3: per determinare lo sconto, dovrò avvalermi dell'elenco dei top10, per sapere se un cliente appartiene a tale elenco, e dovrò sapere qual è stata la spesa di ciascun cliente per gli articoli in promozione
- La lettura/scrittura da/su file è un'operazione molto più onerosa rispetto alla lettura/scrittura da/su tabella di record in memoria centrale
- Non voglio ripetere ogni volta una lettura/scrittura da/su file
- I dati che mi servono appartengono a due categorie: (1) relativi a tutti i clienti, (2) relativi ai top10

- ⇨ posso utilizzare due strutture di supporto (tabelle), interne al programma, su cui andare a scrivere i dati dei file, già predisposti per le operazioni che dovrò svolgere a seconda della scelta dell'utente
- ⇨ i dati che mi serve raccogliere in queste tabelle sono:
  - Per ciascun cliente: ID, Cognome, Spesa nei CD, Spesa nei DVD, Spesa nei CD in Promozione, Spesa nei DVD in Promozione.
  - Per ciascun cliente dei top10: un riferimento alla prima tabella (che contiene i suoi dati).
- ⇨ La tabella top10 può essere semplicemente un vettore di interi: ciascun intero è un indice che punta alla tabella clienti.

Schema funzionale della soluzione proposta



Tipi di dato

Record per leggere il file binario di input (tipo cliente in input: **t_cliente_in**)

- **ID**
- **Cognome**
- **Spesa**
- **Tipo** ('C'/'D')
- **Promozione** ('V'/'F')

Record per la tabella dei clienti (tipo cliente: **t_cliente**)

- **ID**
- **Cognome**
- **Spesa**
- **Spesa_CD** // posso decidere di inserire qui la spesa totale, comprensiva **//** degli articoli in promozione, oppure solamente la spesa **//** relativa agli articoli non in promozione. In questa soluzione, **//** Spesa_CD = spesa totale in CD, compresa la promozione
- **Spesa_DVD**
- **Spesa_CD_Prom**
- **Spesa_DVD_Prom**

Dati

- una tabella **v_clienti** di tipo t_cliente[], di 200 elementi, contenente i dati dei clienti, e un intero **n_clienti** (dimensione logica della tabella).
- una tabella **v_top10** di tipo int[], di 10 elementi, e un intero **n_top10** (dimensione logica della tabella). Ciascun elemento di v_top10 conterrà un indice da 0 a 199, che punta a un elemento di v_clienti.
- puntatori a FILE**: f_clienti**. Il puntatore a file viene utilizzato solo all'interno della funzione crea_tabelle.

Prototipi delle funzioni

```
// crea_tabella_clienti. crea la tabella clienti a partire dal file di input
// riceve in ingresso i nomi dei file di input e l'indirizzo della tabella clienti
// restituisce la dimensione logica della tabella clienti, -1 in caso di errore
int crea_tabella_clienti( char *nome_file_clienti, t_cliente *v_clienti );

// crea_tabella_top10. crea la tabella con gli indici 10 clienti più fedeli a partire dalla
// tabella clienti.
// riceve in ingressol'indirizzo della tabella clienti, la dimensione della tabella
// clienti, e l'indirizzo della tabella top10
// restituisce la dimensione logica della tabella top10, -1 in caso di errore
int crea_tabella_top10( t_cliente *v_clienti, int n_clienti, int *v_top10 );

// visualizza_spesa_cliente. Stampa a video informazioni sul cliente (ID, spesa CD,
// spesa DVD)
// riceve in ingresso una stringa (cognome del cliente) e la tabella clienti
// restituisce l'indice del cliente nella tabella, o -1 se non trovato
int visualizza_spesa_cliente( char *Cognome, t_cliente *v_clienti, int n_clienti );
```

```
// visualizza_top10. visualizza ID, cognome e tot_spesa dei clienti top10
// riceve in input la tabella clienti e la tabella top10
// restituisce il numero di clienti visualizzati, o -1 in caso di errore
int visualizza_top10( t_cliente *v_clienti, int *v_top10, int n_top10 );

// visualizza_top10. visualizza ID, cognome e tot_spesa dei clienti top10
// riceve in input la tabella clienti e la tabella top10
// restituisce il numero di clienti visualizzati, o -1 in caso di errore
int visualizza_sconti( t_cliente *v_clienti, int n_clienti, int *v_top10, int n_top10 );

// menu. Restituisce la scelta dell'utente, e richiama le funzioni.
// riceve in ingresso le tabelle clienti e top10
int menu( t_cliente *v_clienti, int n_clienti, int *v_top10, int n_top10 );

// eventuali funzioni ausiliarie (da determinare al momento del progetto dell'algoritmo
// per queste funzioni)
```

Definizione dei tipi di dato

```
#define SIZE_S 81
#define SIZE_C 200

typedef char t_stringa[SIZE_S];

typedef struct {
        unsigned long ID;
        t_stringa Cognome;
        float Spesa;
        char Tipo; // 'C' o 'D'
        char Promozione; // 'V' o 'F'
} t_cliente_in; // per leggere da clienti.bin

typedef struct {
        unsigned long ID;
        t_stringa Cognome;
        float Spesa_CD; // totale acquisti CD
        float Spesa_DVD; // totale acquisti DVD
        float Spesa_CD_Prom; // acquisti CD in promozione
        float Spesa_DVD_Prom; // acquisti DVD in promozione
} t_cliente; // per eseguire le funzionalità richiamate dal menu
```

Codice del main

```
int main() {
	t_cliente v_clienti[SIZE_C]; // vettore clienti
	int n_clienti=0;

	int v_top10[10]; // gli indici dei 10 clienti che hanno speso di più
	int n_top10=0; // la dimensione logica di v_top10

	n_clienti=crea_tabella_clienti( "clienti.bin", v_clienti );
	n_top10=crea_tabella_top10( v_clienti, n_clienti, v_top10 );
	if( n_clienti<0 )
		return -1;
	while( menu( v_clienti, n_clienti, v_top10, n_top10 ) );

	printf( "Esecuzione terminata\n" );
	return 0;
}
```

Implementazione delle funzioni principali

```
int menu( t_cliente *v_c, int n_c, int *v_t, int n_t ) {
	int scelta;
	t_stringa Cognome;

	printf( "Scegliere... \n ---> [1]: Visualizza cliente\
	\n ---> [2]: Visualizza top10\n ---> [3]: Visualizza sconti\
	\n ---> [0]: Termina\n\n... scelta: " );
	scanf( "%d", &scelta );
	switch (scelta) {
		case 1: printf( "inserire il cognome del cliente... " );
			fflush( stdin );
			gets( Cognome );
			visualizza_spesa_cliente( Cognome, v_c, n_c ); break;
		case 2: visualizza_top10( v_c, v_t, n_t ); break;
		case 3: visualizza_sconti( v_c, n_c, v_t, n_t ); break;
		case 0: break;
		default: printf( "Scelta non riconosciuta.\n" );
	}
	return scelta;
}

int visualizza_top10( t_cliente *v_c, int *v_t, int n_t ) {
	int i;

	for( i=0; i<n_t; i++ )
		printf( "%s [ID: %010lu ] (spesa: %.2f)\n",
			v_c[v_t[i]].Cognome, v_c[v_t[i]].ID,
			v_c[v_t[i]].Spesa_CD_Prom+v_c[v_t[i]].Spesa_DVD_Prom );
	return i;
}
```

```
int visualizza_spesa_cliente( char *s, t_cliente *v, int n ) {
	int i; // indice dell'elemento da visualizzare

	if( ( i=indice( s, v, n ) )<n ) {
		printf( "%s [ID: %010lu ], Spesa CD: %.2f, Spesa DVD: %.2f\n",
			v[i].Cognome, v[i].ID, v[i].Spesa_CD, v[i].Spesa_DVD );
		return i;
	}
	else {
		printf( "Cliente %s non trovato!\n", s );
		return -1;
	}
}

int visualizza_sconti( t_cliente *v_c, int n_c, int *v_t, int n_t ) {
	int i=0, Sconto=0, n_sconto=0;

	for( i=0; i<n_c; i++ ) {
		Sconto= ( top10( i, v_t, n_t ) ) ?
			30 :
			 ( v_c[i].Spesa_CD_Prom+v_c[i].Spesa_DVD_Prom>=200.0 ?
			  20 :
			  2*( ( v_c[i].Spesa_CD_Prom+v_c[i].Spesa_DVD_Prom )/20 )
			 );
		if( Sconto>0 ) {
			printf( "%s [ ID: %010lu ] %2d%%\n",
				v_c[i].Cognome, v_c[i].ID, Sconto );
			n_sconto++;
		}
	}
	return n_sconto;
}
```

ALGORITMI per la creazione delle tabelle, in pseudo-codice

tabella clienti:

```
n_clienti ← 0
apri clienti.bin
per ogni record in clienti.bin
        determina indice in tabella v_clienti a partire da ID
        se non trovato,
                indice ← n_clienti
                aggiorna ID e Cognome
                poni gli altri campi a zero
                incrementa n_clienti
        se Tipo=='C'
                aggiorna Spesa_CD
                se Promozione=='V' aggiorna Spesa_CD_Prom
        se Tipo=='D'
                aggiorna Spesa_DVD
                se Promozione=='V' aggiorna Spesa_DVD_Prom
chiudi clienti.bin
restituisci n_clienti
```

top10:

```
n_top10 ← 0
min ← 0 // indice in top10 del cliente che ha effettuato la spesa minima
per ogni record i in tabella clienti
        se n_top10 < 10
                aggiungi i in v_top10
                fa' sì che min punti all'elemento con la spesa minima
                incrementa n_top10
        altrimenti
                se spesa di i > spesa del cliente individuato da min
                        v_top10[min] ← i
                        fa' sì che min punti all'elemento con la spesa minima
restituisci n_top10
```

```
int crea_tabella_clienti( char *nome_file_clienti, t_cliente *v ) {
	int n=0; // numero di clienti creati = dimensione logica della tabella clienti
	t_cliente_in temp;
	FILE *f_clienti, *f_acquisti;
	int i; // indice in tabella clienti

	if( !( f_clienti=fopen( nome_file_clienti, "rb" ) ) )
		exit( -1 );

	while( fread( &temp, sizeof( t_cliente_in ), 1, f_clienti ) ) {
		i=indice_ID( temp.ID, v, n );
		if( !( i<n ) ) { // se nuovo cliente
			// i=n;
			v[i].ID=temp.ID;
		   strcpy( v[i].Cognome, temp.Cognome );
			// valore iniziale dei campi ← 0.0
			v[i].Spesa_DVD=0.0;
			v[i].Spesa_CD=0.0;
			v[i].Spesa_DVD_Prom=0.0;
			v[i].Spesa_CD_Prom=0.0;
			n++;
		}
		if( temp.Tipo=='C' ) {
			v[i].Spesa_CD+=temp.Spesa;
			if( temp.Promozione=='V' )
				v[i].Spesa_CD_Prom+=temp.Spesa;
		}
		else {
			v[i].Spesa_DVD+=temp.Spesa;
			if( temp.Promozione=='V' )
				v[i].Spesa_DVD_Prom+=temp.Spesa;
		}
	}

	fclose( f_clienti );

	return n;
}
```

```
int crea_tabella_top10( t_cliente *v_c, int n_c, int *v_t ) {
        int n_t=0; // dimensione logica della tabella top10
        int min=0;  // indice in top10 del cliente che ha effettuato la spesa minima
        int i; // indice nella tabella clienti
        int j; // indice nella tabella top10

        for( i=0; i<n_c; i++ ) {
                if( v_c[i].Spesa_CD_Prom+v_c[i].Spesa_DVD_Prom==0.0 )
                        continue;
                if( n_t<10 ) {
                        v_t[n_t]=i;
                        if(     v_c[i].Spesa_CD_Prom
                                        +v_c[i].Spesa_DVD_Prom
                                        < v_c[v_t[min]].Spesa_CD_Prom
                                        +v_c[v_t[min]].Spesa_DVD_Prom )
                                min=n_t;
                        n_t++;
                }
                else {
                        if(     v_c[i].Spesa_CD_Prom
                                        +v_c[i].Spesa_DVD_Prom
                                        > v_c[v_t[min]].Spesa_CD_Prom
                                        + v_c[v_t[min]].Spesa_DVD_Prom ) {
                                v_t[min]=i;
                                min=0;
                                for( j=1; j<n_t; j++ ) {
                                        if(     v_c[v_t[j]].Spesa_CD_Prom
                                             + v_c[v_t[j]].Spesa_DVD_Prom
                                             < v_c[v_t[min]].Spesa_CD_Prom
                                             + v_c[v_t[min]].Spesa_DVD_Prom )
                                                min=j;
                                }
                        }
                }
        }
        return n_t;
}
```

Implementazione delle funzioni ausiliarie

```
// indice.
// restituisce l'indice di un cliente nella tabella clienti, a partire dal cognome
// se non trovato, restituisce la dimensione della tabella
int indice( char *Cognome, t_cliente *v, int dim ) {
	int i;
	for( i=0; i<dim; i++ )
		if( !strcmp( v[i].Cognome, Cognome ) )
			break;
	return i;

}

// indice_ID.
// restituisce l'indice di un cliente nella tabella clienti, a partire dall'ID
// se non trovato, restituisce la dimensione della tabella
int indice_ID( unsigned long ID, t_cliente *v, int dim ) {
	int i;
	for( i=0; i<dim; i++ )
		if( v[i].ID==ID )
			break;
	return i;

}

// top10. restuisce 1 se il cliente è in top10, 0 altrimenti
int top10( int i, int *v_t, int n_t ) {
	int j;
	for( j=0; j<n_t; j++ )
		if( v_t[j]==i )
			return 1;
	return 0;
}
```