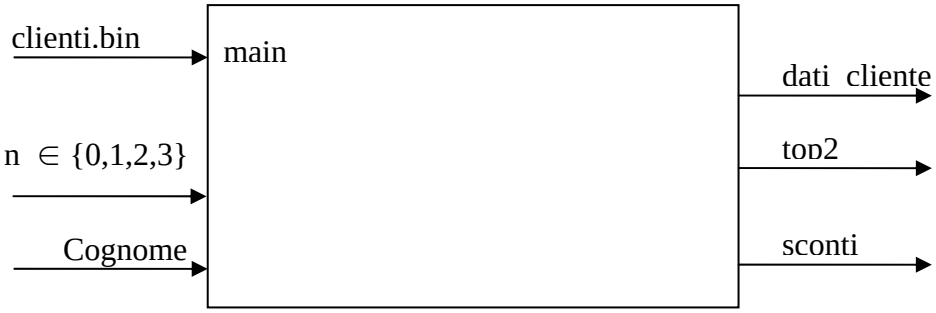
Fondamenti di Informatica L-A (Elettronica/Automazione)
A.A. 2005/2006, quarto scritto (5/4/2005)
Proposta di soluzione (ragionata) dell'esercizio di progetto[1]

Input/Output
Input = clienti.bin, n (funzione richiesta dall'utente), Cognome (per il punto 1)
Output = dati_cliente, top2, sconti



Funzionalità richieste

1. Visualizzazione di ID e giorni trascorsi da un cliente. Mi servono:
   - il cognome del cliente
   - la somma della spesa del cliente in bassa stagione
   - la somma della spesa del cliente in alta stagione
2. Visualizzazione dei 2 clienti più fedeli ("top2"). Mi servono:
   - 2 riferimenti a clienti
   - i giorni trascorsi in bassa stagione da ciascun cliente
3. Visualizzazione dello sconto. Mi servono:
   - i giorni trascorsi in bassa stagione da ciascun cliente
   - l'elenco dei top2 (punto 2)

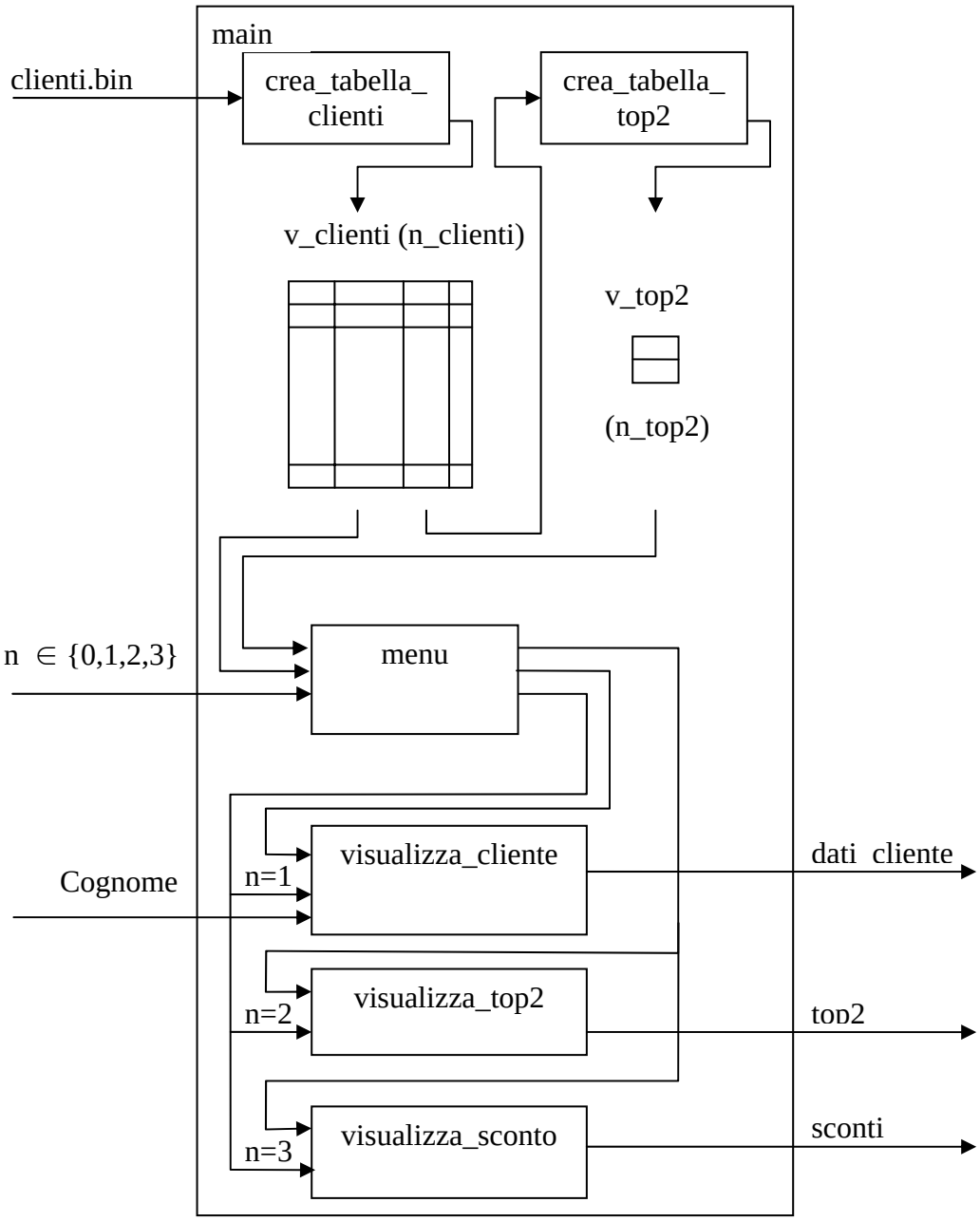
Considerazioni iniziali di progetto

- 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 "top2" dovrò selezionare alcuni tra i clienti
  - ⇨ nel punto 3: per determinare lo sconto, dovrò sapere chi sono i top2, e dovrò sapere qual è stato il numero di giorni trascorsi da ciascun cliente in bassa stagione
- 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 top2
  - ⇨ posso utilizzare una struttura di supporto (tabella), interna 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

[1] **Consiglio: vedi anche la soluzione dell'esercizio dato il 18/1/2006**

- ⇨ per quanto riguarda i top2, posso riusare la soluzione l'algoritmo proposto con la soluzione dell'esercizio del 18/1/2006 (con una tabella di 2 interi), oppure inventare un nuovo algoritmo.
- ⇨ i dati che mi serve raccogliere in tabella sono:
  - Per ciascun cliente: ID, Cognome, Giorni trascorsi in b.s., Giorni trascorsi in a.s..
  - Per ciascun cliente dei top2: un riferimento alla prima tabella (che contiene i suoi dati).

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**
- **Giorni**
- **Tipo** (“A1”/”B5”/…)
- **BassaStagione** (‘V’/‘F’)

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

- **ID**
- **Cognome**
- **Giorni_BS**
- **Giorni_AS** // In questa soluzione, il numero totale di giorni trascorsi nello // stabilimento è pari a Giorni_AS + Giorni_BS

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_top2** di tipo int[], di 2 elementi, e un intero **n_top2** (dimensione logica della tabella). Ciascun elemento di v_top2 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_top2. crea la tabella con gli indici dei 2 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 top2
// restituisce la dimensione logica della tabella top2, -1 in caso di errore
int crea_tabella_top2( t_cliente *v_clienti, int n_clienti, int *v_top2 );

// visualizza_giorni_cliente. Stampa a video informazioni sul cliente (ID, giorni a.s.,
// giorni b.s.)
// 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_giorni_cliente( char *Cognome, t_cliente *v_clienti, int n_clienti );

// visualizza_top2. visualizza ID, cognome e tot_giorni dei due clienti più fedeli
// riceve in input la tabella clienti e la tabella top2
// restituisce il numero di clienti visualizzati, o -1 in caso di errore
int visualizza_top2( t_cliente *v_clienti, int *v_top2, int n_top2 );

// visualizza_sconti. visualizza ID, cognome e sconti dei clienti
```

```
// riceve in input la tabella clienti e la tabella top2
// restituisce il numero di clienti visualizzati, o -1 in caso di errore
int visualizza_sconti( t_cliente *v_clienti, int n_clienti, int *v_top2, int n_top2 );

// menu. Restituisce la scelta dell'utente, e richiama le funzioni.
// riceve in ingresso le tabelle clienti e top2
int menu( t_cliente *v_clienti, int n_clienti, int *v_top2, int n_top2 );

// 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;
        int Giorni;
        char Pacchetto[3]; // "A5"/"B2"/…
        char BassaStagione; // 'V' o 'F'
} t_cliente_in; // per leggere da clienti.bin

typedef struct {
        unsigned long ID;
        t_stringa Cognome;
        int Giorni_AS; // totale giorni in alta stagione
        int Giorni_BS; // totale giorni in bassa stagione
} 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_top2[2]; // gli indici dei 2 clienti che hanno speso di più
	int n_top2=0; // la dimensione logica di v_top2

	n_clienti=crea_tabella_clienti( "clienti.bin", v_clienti );
	if( n_clienti<0 )
		return -1;
	n_top2=crea_tabella_top2( v_clienti, n_clienti, v_top2 );
	while( menu( v_clienti, n_clienti, v_top2, n_top2 ) );

	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 2 clienti piu' fedeli\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_giorni_cliente( Cognome, v_c, n_c ); break;
		case 2: visualizza_top2( 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_top2( t_cliente *v_c, int *v_t, int n_t ) {
	int i;

	for( i=0; i<n_t; i++ )
		printf( "%s [ID: %010lu ] (giorni b.s.: %d)\n",
			v_c[v_t[i]].Cognome, v_c[v_t[i]].ID,
			v_c[v_t[i]]. Giorni_BS );
	return i;
}
```

```
int visualizza_giorni_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 ], giorni in a.s.: %d, giorni in b.s.: %d\n",
			v[i].Cognome, v[i].ID, v[i].Giorni_AS, v[i].Giorni_BS );
		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= ( top2( i, v_t, n_t ) ) ?
			30 :
			 ( v_c[i].Giorni_BS>=20 ?
			  20 :
			  2*( ( v_c[i].Giorni_BS )/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 BassaStagione=='V'
                aggiorna Giorni_BS
        altrimenti
                aggiorna Giorni_AS
chiudi clienti.bin
restituisci n_clienti
```

top2:

```
se n_clienti = 0 restitusci n_top2 ← 0 ed esci
se n_clienti = 1 restitusci n_top2 ← 1, v_top2[0] ← 0 ed esci
altrimenti
        n_top2 ← 2,
        v_top2[0] ← 0
        v_top2[1] ← 1
        per ogni record i in tabella clienti, a partire dal terzo, fino all'ultimo
                se v[i] ha più giorni in b.s. di quelli di v_top2[0]
                        v_top2[1] ← v_top2[0]
                        v_top2[0] ← i
                altrimenti, se v[i] ha più giorni in b.s. di quelli di v_top2[1]
                        v_top2[1] ← i
restituisci n_top2
```

```
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;
        int i; // indice in tabella clienti

        if( !( f_clienti=fopen( nome_file_clienti, "rb" ) ) )
                return -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
                        v[i].Giorni_BS=0;
                        v[i].Giorni_AS=0;
                        n++;
                }
                if( temp.BassaStagione=='V' )
                        v[i].Giorni_BS+=temp.Giorni;
                else
                        v[i].Giorni_AS+=temp.Giorni;
        }

        fclose( f_clienti );

        return n;
}
```

```
int crea_tabella_top2( t_cliente *v_c, int n_c, int *v_t ) {
	int n_t; // dimensione logica della tabella top10
	int i; // indice nella tabella clienti

	switch( n_c ) {
		case 0 : n_t=0; break;
		case 1 : n_t=1; v_t[0] = 0; break;
		default : n_t=2;
			if( v_c[0].Giorni_BS > v_c[1].Giorni_BS ) {
				v_t[0] = 0;
				v_t[1] = 1;
			}
			else {
				v_t[0] = 1;
				v_t[1] = 0;
			}
			for( i=2; i<n_c; i++ ) {
				if( v_c[i].Giorni_BS > v_c[v_t[0]].Giorni_BS ) {
					v_t[1] = v_t[0];
					v_t[0] = i;
				}
				else if( v_c[i].Giorni_BS > v_c[v_t[1]].Giorni_BS ) {
					v_t[1] = i;
				}
			}
	}

	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;

}

// top2. restuisce 1 se il cliente è in top2, 0 altrimenti
int top2( 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;
}
```