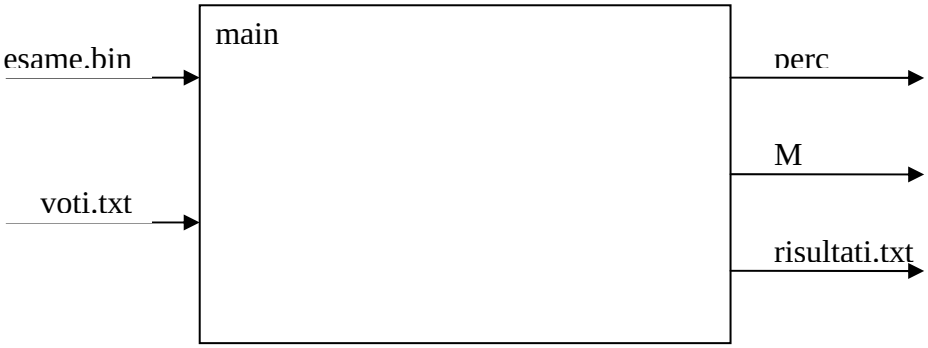
Fondamenti di Informatica L-A (Elettronica/Automazione)
A.A. 2005/2006, secondo parziale (16/12/2005)
Proposta di soluzione (ragionata) dell'esercizio di progetto B/D

Input/Output
Input = esame.bin, voti.txt
Output = perc, M, risultati.txt



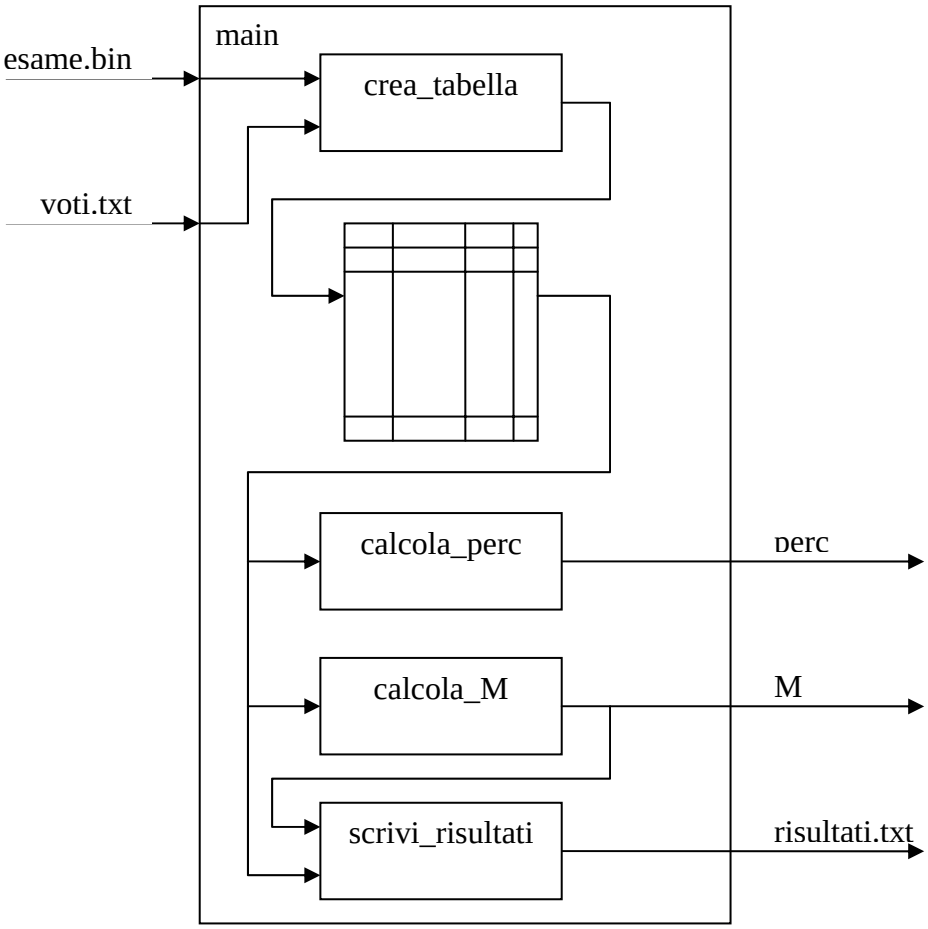
Operazioni

1. determinazione della percentuale di studenti che hanno superato l'esame al primo tentativo (perc). Mi servono:
   - il numero di studenti totali (= numero di record nei file di input)
   - il numero di studenti che hanno superato al primo tentativo (lo ricavo contando quanti studenti in voti.txt hanno il campo Respinto = 0)
2. determinazione di M. Mi servono:
   - la media degli studenti che hanno passato al primo tentativo (dal file voti.txt)
   - la media degli studenti respinti almento una volta (dal file voti.txt)
3. scrittura su risultati.txt. Mi servono:
   - Matricola, cognome (da esame.bin), e voto (da voti.txt)
   - Il risultato del punto 2 (M)

Considerazioni iniziali di progetto

- Il file risultati.txt (punto 3) deve contenere dati provenienti sia da esame.bin che da voti.txt, opportunamente combinati
- La lettura/scrittura da/su file è un'operazione molto più onerosa rispetto alla lettura/scrittura da/su tabella di record in memoria centrale
- I dati contenuti nei file mi servono per varie operazioni: calcolo della percentuale, di M, produzione di risultati.txt: non voglio ripetere ogni volta una lettura/scrittura da/su file
  - ⇨ 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
  - ⇨ i dati che mi serve raccogliere in questa tabella sono: Matricola, Cognome, Voto (per il punto 3) e Respinto (per i punti 1 e 2)

Schema funzionale della soluzione proposta



Tipi di dato

Record per leggere esame.bin (tipo esame: **t_esame**)

- **Matricola**
- **Cognome**
- **Nome**

Record per la tabella (tipo studente: **t_studente**)

- **Matricola**
- **Cognome**
- **Voto**
- **Respinto**

Dati

- un record **r_esam** di tipo t_esame per leggere da esame.bin. r_esam viene utilizzato solo all'interno della funzione crea_tabella
- una tabella **v_stud** di tipo t_studente[], e un intero **n_stud** (dimensione della tabella). v_stud e n_stud vengono utilizzati da tutte le funzioni
- due interi, **perc** ed **M** utilizzati dal main e dalla funzione scrivi_risultato (M)
- tre puntatori a FILE**: f_esam**, **f_voti**, **f_risu**. I puntatori a file vengono utilizzati solo all'interno delle rispettive funzioni (crea_tabella e scrivi_risultato)

Prototipi delle funzioni

```
// crea_tabella. input: esame.bin e voti.txt. Output: v_stud e n_stud
// restituisce il puntatore alla tabella creata. La dimensione della tabella è un secondo
// output: ricorro al passaggio per riferimento (potevo usare invece delle variabili
// globali). In caso di fallimento restituisce NULL.
t_studente *crea_tabella( char* nome_file_esame, char *nome_file_voti, int *dim );

// calcola_perc. input: v_stud e n_stud. Output: perc.
int calcola_perc( t_studente * v, int dim );

// calcola_M. input: v_stud e n_stud. Output: M.
int calcola_M( t_studente * v, int dim );

// scrivi_risultato. input: v_stud, n_stud ed M. Output: risultati.txt.
// Restituisce 0 in caso di successo,  -1 in caso di errori (es. problemi di apertura file)
int scrivi_risultato( t_studente * v, int dim, int media, char* nome_file_risultati );

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

Definizione dei tipi di dato

```
typedef  struct {
        unsigned long Matricola; // sufficiente per contenere una matricola
        char Cognome[51]; // attribuisco un valore (non è specificato nel testo)
        char Nome[51];
} t_esame;

typedef struct {
        unsigned long Matricola; // lo stesso tipo usato in t_esame
        char Cognome[51]; // lo stesso tipo usato in t_esame
        char Voto; // un intero da 0 a 30 sta in un char (potevo usare comunque int)
        char Respinto; // valori: 0/1. Vedi sopra.
} t_studente;
```

Codice del main

```
int main() {
        int perc, M, n_stud;
        t_studente *v_stud;

        v_stud=crea_tabella( "esame.bin", "voti.txt", &n_stud );
        if( v_stud==NULL )
                exit( -1 );
        perc=calcola_perc( v_stud, n_stud );
        printf( "%d %%\n",  perc );
        M = calcola_M(v_stud, n_stud );
        printf( "%d\n",  M );
        scrivi_risultato( v_stud, n_stud, M, "risultati.txt" );
        free( v_stud );
}
```

Implementazione delle funzioni principali

```
int calcola_perc( t_studente * v, int dim ) {
	int i, p=0;
	for( i=0; i<dim; i++ ) {
		if( v[i].Respinto==0 )
			p++;
	}
	return 100*p/dim;
}

int calcola_M( t_studente * v, int dim ) {
	int i, M_primo=0, n_primo=0, M_resp=0, n_resp=0;
	for( i=0; i<dim; i++ ) {
		if( v[i].Respinto==0 ) {
			M_primo += v[i].Voto;
			n_primo++;
		}
		else{
			M_resp += v[i].Voto;
			n_resp++;
		}

	}
	M_primo = n_primo>0 ? M_primo/n_primo : 0;
	M_resp = n_resp>0 ? M_resp/n_resp : 0;
	return M_primo > M_resp ? M_primo : M_resp;
}

int scrivi_risultato( t_studente * v, int dim, int media, char* nome_file_risultati ) {
	int i;
	FILE *f_risu;

	f_risu=fopen( nome_file_risultati, “w” );
	if( f_risu==NULL )
		return -1;

	for( i=0; i<dim; i++ ) {
		if( v[i].Voto>=media )
			fprintf( f_risu, “%lu %s %d\n”,
				v[i].Matricola,
				v[i].Cognome,
				v[i].Voto );
	}

	fclose( f_risu );
	return 0;
}
```

```
t_studente *crea_tabella( char* nome_file_esame, char *nome_file_voti, int *dim ) {
        int i;
        FILE *f_esam, *f_voti;
        t_studente *v;
        t_esame r_esame;
        unsigned long Matricola; // per la lettura da voti.txt
        int Voto, Respinto, Giorno; // per la lettura da voti.txt

        // creazione della tabella e lettura da esame.bin
        f_esam=fopen(nome_file_esame, "rb" );
        if( f_esam==NULL )
                return NULL;

        // crea tabella come dato dinamico
        *dim=dimensione( f_esam, sizeof( t_esame ) );
        v=( t_studente* )malloc( *dim*sizeof( t_esame ) );
        if( v==NULL )
                return NULL;

        // ciclo di lettura da esame.bin
        *dim=0;
        while( fread( &r_esame, sizeof( r_esame ), 1, f_esam )>0 ) {
                // servono Matricola e Cognome
                v[*dim].Matricola=r_esame.Matricola;
                strcpy( v[*dim].Cognome,r_esame.Cognome );
                ( *dim )++;
        }

        fclose( f_esam );

        // lettura da voti.txt
        f_voti=fopen(nome_file_voti, "r" );
        if( f_voti==NULL )
                return NULL;

        while( fscanf( f_voti, "%lu %d %d %d",
                 &Matricola, &Voto, &Giorno, &Respinto )>0 ) {
                i=indice( Matricola, v, *dim );
                if( i<*dim ) {
                        v[i].Voto=Voto;
                        v[i].Respinto=Respinto;
                }
        }

        fclose( f_voti );

        return v;
}
```

Implementazione delle funzioni ausiliarie

```
// determina il numero di record in un file f, data la lunghezza del record.
// Posiziona (come side effect) il puntatore a file all'inizio del file
int  dimensione( FILE* f, int s ) {
        int dim;
        fseek( f, 0, SEEK_END );
        dim = ftell( f )/s;
        fseek( f, 0, SEEK_SET );
        return dim;
}

// determina l'indice di un record all'interno del vettore v, data la Matricola.
// prevedo che in voti.txt possa essere presente (magari per errore) una matricola
// non contenuta in esami.bin: in questo caso, indice restituisce dim.
int indice( int Matricola, t_studente *v, int dim ) {
        int i;
        for( i=0; i<dim; i++ )
                if( v[i].Matricola==Matricola )
                        break;
        return i;
}
```

Considerazioni sull'algoritmo di creazione della tabella

Una soluzione alternativa poteva essere quella di considerare a uno a uno i record letti da esame.bin, e quindi cercare il record corrispondente in voti.txt, tramite un ciclo di lettura da file. Considerando che per trovare il primo record è necessaria una lettura, e per l'ultimo sono necessarie n_stud letture da voti.txt, tramite scanf, il numero di letture complessive di questa soluzione è dell'ordine di n_stud^2.

Nell'ottica di ridurre il più possibile le letture a file, è preferibile la soluzione proposta in quanto prevede un numero di letture dell'ordine di n_stud.