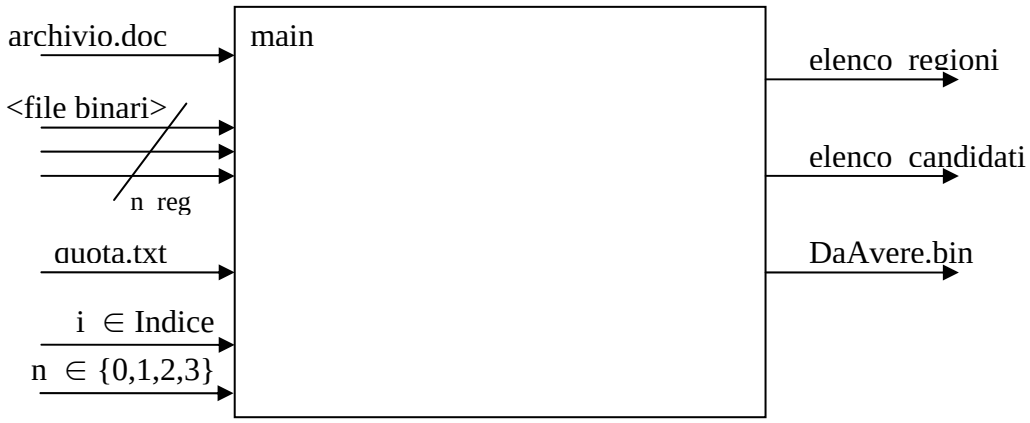
Fondamenti di Informatica L-A (Elettronica/Automazione)
A.A. 2005/2006, secondo scritto (21/12/2005)
Proposta di soluzione (ragionata) dell'esercizio di progetto

Input/Output
Input = archivio.doc, file binari elencati in archivio.doc, quota.txt, i (indice della regione), n (funzione richiesta dall'utente)
Output = elenco_regioni, elenco_candidati, DaAvere.bin



Funzionalità richieste

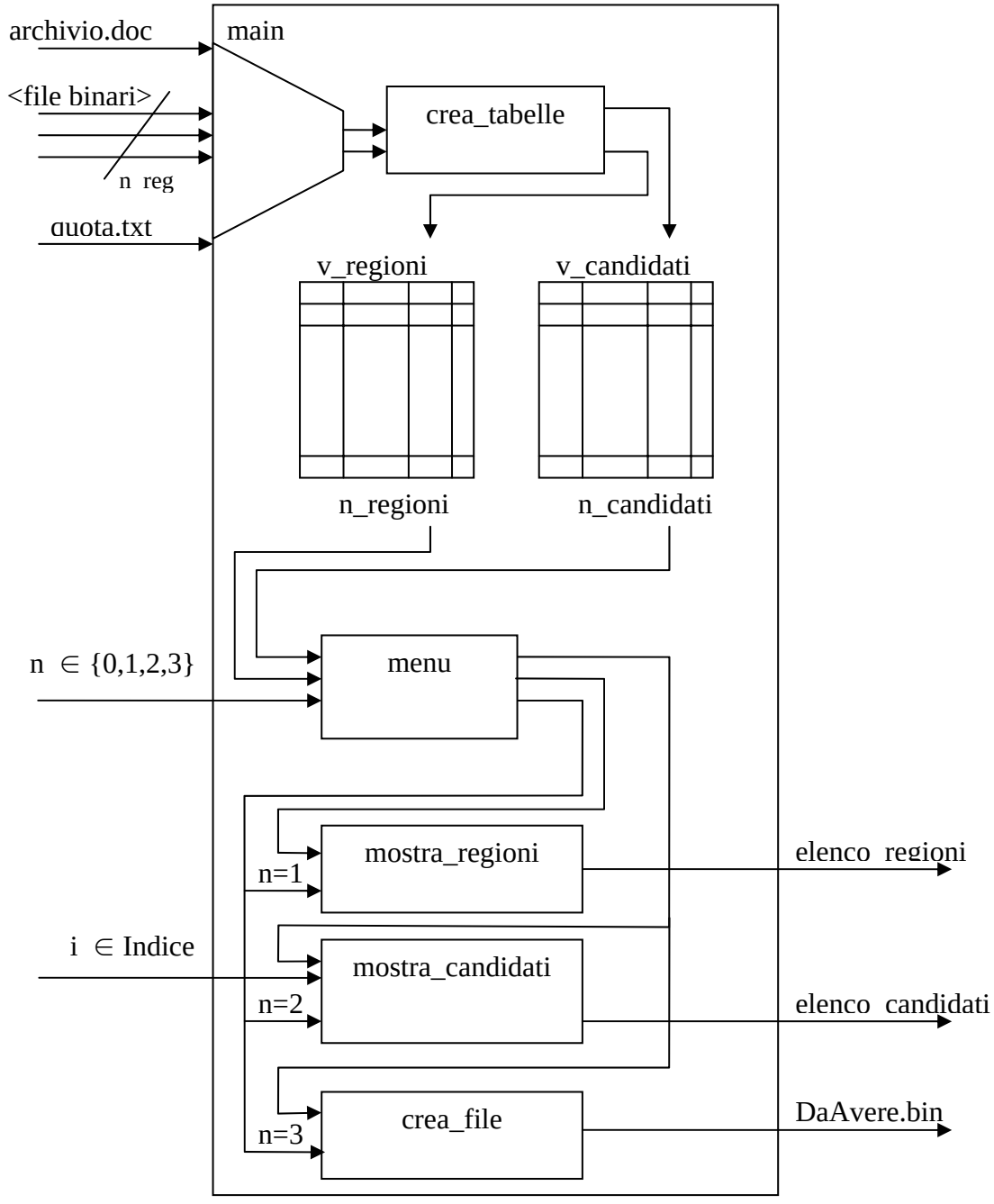
1. Visualizzazione dell'elenco delle regioni. Mi servono:
   - il numero di regioni totali (= numero di righe nel file archivio.doc)
   - il numero di candidati per ciascuna regione (= numero di record di ciascun file associato a ciascuna regione)
   - il totale delle quote versate su ciascuna regione (= totale dei valori Quota relativi a ciascun candidato, dal file quota.txt)
2. Visualizzazione dell'elenco dei candidati. Mi servono:
   - il numero totale dei candidati (somma dei valori al punto 1)
   - l'indice di ciascuna regione (come da punto 1)
   - i dati Cognome, Telefono e Data di ciascun candidato di ciascuna regione, contenuti nel file binario relativo alla regione
3. scrittura su DaAvere.bin. Mi servono:
   - Codice, Cognome, Telefono, Data (dai file binari delle regioni), e Quota_Dovuta (dal file quota.txt)

Considerazioni iniziali di progetto

- Le informazioni necessarie per visualizzare l'elenco delle regioni richiedono dati provenienti da tutti i file. Simili considerazioni valgono per gli alti punti.
- 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 dei totali delle quote, delle quote da versare, produzione di DaAvere.bin: non voglio ripetere ogni volta una lettura/scrittura da/su file
- I dati che mi servono appartengono a due categorie: raggruppati per regioni, oppure relativi a ciascun singolo candidato

- ⇨ 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 ciascuna regione: Regione (nome della regione), N_Cand (numero di candidati), Tot_Q (totale delle quote versate), per il punto 1
  - Per ciascun candidato: Codice, Cognome, Telefono, Data, Indice della regione di appartenenza, Quota_D (quota ancora dovuta), per i punti 2 e 3.

Schema funzionale della soluzione proposta



Tipi di dato

Record per leggere i file binari delle regioni (tipo candidato in input: **t_cand_in**)

- **Codice**
- **Cognome**
- **Telefono**
- **Data** (sarà necessario definire anche un tipo **t_data**)

Record per la tabella delle regioni (tipo regione: **t_regione**)

- **Regione**
- **N_Cand**
- **Tot_Q**

Record per la tabella dei candidati (tipo candidato: **t_candidato**)

- **Codice**
- **Cognome**
- **Telefono**
- **Data**
- **Regione**
- **Quota** (quota versata)

Record per scrivere sul file binario DaAvere (tipo candidato in output: **t_cand_out**)

- **Codice**
- **Cognome**
- **Telefono**
- **Data**
- **Quota** (quota dovuta)

Dati

- una tabella **v_regioni** di tipo t_regione[] contenente i dati delle regioni, e un intero **n_regioni** (dimensione della tabella). L'indice della regione può corrispondere all'indice della tabella (eventualmente +1, se si vuole partire da 1 anziché da 0). v_regioni e n_regioni vengono inizializzati da crea_tabelle e devono essere usati da mostra_regioni.
- una tabella **v_candidati** di tipo t_candidato[], e un intero **n_ candidati** (dimensione della tabella). v_candidati e n_candidati vengono inizializzati da crea_tabelle e devono essere usati da mostra_candidati e crea_file.
- puntatori a FILE**: f_archivio**, **f_regione**, **f_quota**, **f_avere**. I puntatori a file vengono utilizzati solo all'interno delle rispettive funzioni (crea_tabelle e crea_file).

Scelte progettuali per la visibilità delle variabili

Bisogna scegliere se considerare le tabelle v_regioni e v_candidati come variabili locali o come dati globali. Nel primo caso, siccome la funzione crea_tabelle restituisce quattro valori (due coppie tabella/dimensione: la dimensione delle tabelle non è nota a priori), il prototipo della funzione crea_tabelle potrebbe essere del tipo:

```
int crea_tabelle(   char* nome_file_archivio, char* nome_file_quota,
                    t_regione **punt_v_regioni, int *punt_n_regioni,
                    t_candidato **punt_v_candidati, int *punt_n_candidati    );
```

Ricorrendo a variabili globali invece, si potrebbe pensare a una funzione del tipo:

```
int crea_tabelle( void );
```

in cui tutti i dati sono globali, o staticamente assegnati (compresi i nomi dei file, ad esempio). Proporremo una soluzione senza variabili globali.

Prototipi delle funzioni

```
// crea_tabella. input: nomi dei file archivio.doc e quota.txt.
// Output: v_regioni, n_regioni, v_candidati e n_candidati. L'output viene restituito
// tramite parametri passati per riferimento. La funzione restituisce 0 in caso di
// successo, -1 in caso di fallimento.
int crea_tabelle(       char* nome_file_archivio, char* nome_file_quota,
                        t_regione **punt_v_regioni, int *punt_n_regioni,
                        t_candidato **punt_v_candidati, int *punt_n_candidati       );

// mostra_regioni. input: v_regioni e n_regioni.
// Restituisce il numero di regioni visualizzate.
int mostra_regioni( t_regione *v, int dim );

// mostra_candidati. input: v_candidati, n_candidati, e indice (della regione).
// Restituisce il numero di candidati visualizzati.
int mostra_candidati( t_candidato *v, int dim, int reg );

// crea_file. input: v_candidati, n_candidati e nome del file di output.
// Restituisce 0 in caso di successo,  -1 in caso di errori (es. problemi di apertura file)
int crea_file( t_candidato *v, int dim, char* nome_file_avere );

// menu. input: v_regioni, n_regioni, v_candidati, n_candidati.
// Restituisce il numero associato alla funzione richiesta (1/2/3 o 0 per terminare)
int menu( t_regione *v_r, int dim_r, t_candidato *v_c, int dim_c );

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

Definizione dei tipi di dato

```
typedef struct {
        unsigned char Giorno; // ad esempio (vanno bene anche altri tipi interi)
        unsigned char Mese;
        unsigned short int Anno;
} t_data;

typedef  struct {
        unsigned long Codice; // o qualsiasi altro tipo intero
        char Cognome[51]; // attribuisco un valore (non è specificato nel testo)
        char Telefono[21]; // idem
        t_data Data;
} t_cand_in;

typedef struct {
        char Regione[51];
        int N_Cand;
        float Tot_Q;
} t_regione;
```

```
typedef struct {
	unsigned long Codice;
	char Cognome[51];
	char Telefono[21];
	t_data Data;
	unsigned char Regione; // indice nella tabella delle regioni (max 20)
	float Quota; // quota versata
} t_candidato;

typedef struct {
	unsigned long Codice;
	char Cognome[51];
	char Telefono[21];
	t_data Data;
	float Quota; // quota dovuta
} t_cand_out;
```

Codice del main

```
int main() {
	t_regione *v_regioni;
	t_candidato *v_candidati;
	int n_regioni, n_candidati, result;

	result=crea_tabelle(	“archivio.doc”, “quota.txt”,
				&v_regioni, &n_regioni,
				&v_candidati, &n_candidati  );
	if( result<0 ) exit( -1 );
	while( menu( v_regioni, n_regioni, v_candidati, n_candidati ) );

	free( v_regioni );
	free( v_candidati );
}
```

Implementazione delle funzioni principali

```
int menu( t_regione *v_r, int dim_r, t_candidato *v_c, int dim_c ) {
	int scelta, r;

	printf( "Scegliere [1/2/3] o 0 per uscire... " );
	scanf( "%d", &scelta );
	switch (scelta) {
		case 1: mostra_ regioni( v_r, dim_r ); break;
		case 2: printf( "selezionare una regione [1..%d]... ", dim_r );
			scanf( "%d", &r );
			if( ( r<1 )||( r> dim_r ) )
				printf( "%d: Scelta non riconosciuta.
					Inserire un numero da 1 a %d\n", r, dim_r );
			else
				mostra_candidati( v_c, dim_c, r-1 ); break;
	case 3: crea_file( v_c, dim_c, “DaAvere.bin” ); break;
		case 0: break;
		default: printf( "Scelta non riconosciuta.\n" );
	}
	return scelta;
}

int mostra_regioni( t_regione *v, int dim ) {
	int r;

	printf( "Selezionato 1: Elenco regioni\n" );
	for( r=0; r<dim; r++ )
		printf( "%d - Regione %s:\t%2d candidati,\tTOT Quote: %.2f\n",
			r+1, v[r].Regione, v[r].N_Cand, v[r].Tot_Q );

	return r;
}

int mostra_candidati( t_candidato *v, int dim, int r ) {
	int i;

	printf( "Selezionato 2: Candidati regione N.%d\n", r+1 );
	for( i=0; i<dim; i++ ) {
		if( v[i].Regione!=r ) continue;
		printf( "Codice: %010ul; Cognome: %s; Tel.: %s;
			Nata/o il: %02d/%02d/%04d\n",
				v[i].Codice,
				v[i].Cognome,
				v[i].Telefono,
				v[i].Data.Giorno,
				v[i].Data.Mese,
				v[i].Data.Anno );
	}
	return i;
}
```

```
int crea_file( t_candidato *v, int dim, char* nome_file_avere ) {
	int i, scritti=0;
	FILE *f_avere;
	t_cand_out temp;

	printf( "Selezionato 3: Creazione file di output (%s)\n", nome_file_avere );
	if( !(f_avere=fopen( nome_file_avere, "wb" ) ) ) {
		printf( "Errore in apertura file %s", nome_file_avere );
		return 0;
	}
		for( i=0; i<dim; i++ ) {
			if( v [i].Quota<100.00 ) {
				strcpy( temp.Cognome, v[i]. Cognome );
				strcpy( temp.Telefono, v[i]. Telefono );
				temp.Quota=100.0-v[i].Quota;
				fwrite( &temp, sizeof( t_cand_out ), 1, f_avere );
				printf( "Scritto: %s; Tel.: %s; Da Avere: %.2f\n",
					temp.Cognome, temp.Telefono, temp.Quota );
				scritti++;
			}
		}
		fclose( f_avere );

		printf( "Scritti %d record su %s\n", scritti, nome_file_avere );
	return scritti;
}
```

/* ALGORITMO per la creazione delle tabelle, in pseudo-codice

```
int crea_tabelle(	char* nome_file_archivio, char* nome_file_quota,
			t_regione **punt_v_regioni, int *punt_n_regioni,
			t_candidato **punt_v_candidati, int *punt_n_candidati	) {

// [1] Dimensionamento delle tabelle
// [2] Creazione delle tabelle tramite malloc
// [3] Immissione dei dati delle regioni
// [4] Immissione dei dati delle quote
// [5] Restituzione risultato (0 in caso di successo)
}
```

[1] Dimensionamento delle tabelle

```
apertura file archivio.doc ( f_archivio )
contatori numero regioni e numero candidati=0
while( lettura di una regione )
	aggiornamento del contatore delle regioni
	apertura del file binario corrispondente ( f_regione )
	determinazione del numero di record
	aggiornamento del contatore del numero di candidati
	chiusura di f_regione
(fine while)
rewind di f_archivio
```

[2] Creazione delle tabelle tramite malloc
Nota: in alternativa, si sarebbe potuta ipotizzare una dimensione massima della tabella delle regioni (v_regioni) di 20 record: e in questo modo definire v_regioni come variabile non dinamica.

[3] Immissione dei dati delle regioni
```
indice_r = 0, indice_c=0
while( lettura di una regione )
        aggiornamento del campo Regione di v_regioni[indice_r]
        apertura del file binario corrispondente ( f_regione )
        v_regioni[indice_r].N_Cand = 0
        per ogni record,
                incremento del campo N_Cand di v_regioni[indice_r]
                aggiornamento dei campi Codice, Conome, Telefono e Data
                di v_candidati[indice_c] e del campo Regione (= indice_r)
                incremento di indice_c
        chiusura di f_regione
        incremento di indice_r
(fine while)
chiusura di f_archivio
```

[4] Immissione dei dati delle quote
```
apertura file quota.txt ( f_quota )
while( lettura di una quota )
        individuazione del corrispondente candidato (indice_c)
        aggiornamento di v_candidati[indice_c].Quota
        aggiornamento di v_regioni[v_candidati[indice_c].Regione].Quota
(fine while)
chiusura di f_quota
*/

int crea_tabelle(       char* nome_file_archivio, char* nome_file_quota,
                        t_regione **punt_v_regioni, int *punt_n_regioni,
                        t_candidato **punt_v_candidati, int *punt_n_candidati       ) {
        FILE *f_archivio, *f_regione, *f_quota;
        int indice_r=0, indice_c=0;
        char Regione[51], NomeFile[121]; // per la lettura da archivio.doc
        t_regione *v_r; // per alleggerire il codice ( per i punti 3 e 4 );
        t_candidato *v_c; // idem
        int n_r, n_c; // idem
        t_cand_in temp; // per la lettura dai file binari delle regioni
        unsigned long Codice; float Quota; // per la lettura da quota.txt

        // [1] Dimensionamento delle tabelle
        if( !(f_archivio=fopen( nome_file_archivio, "r" ) ) ) {
                printf( "Errore in apertura file %s\n", nome_file_archivio );
                return -1;
        }
        *punt_n_regioni=0;
```

```
*punt_n_candidati=0;
while( fscanf( f_archivio, "%s %s", Regione, NomeFile )>0 ) {
        ( *punt_n_regioni )++;
        if( !( f_regione=fopen( NomeFile, “rb” )  )  ) {
                printf( "Errore in apertura file %s\n", NomeFile );
                fclose( f_archivio );
                return -1;
        }
        fseek( f_regione, 0, SEEK_END );
        ( *punt_n_candidati ) += ftell( f_regione ) / sizeof( t_cand_in );
        fclose( f_regione );
} // ( fine while )
n_r=*punt_n_regioni;
n_c=*punt_n_candidati;
rewind( f_archivio );

// [2] Creazione delle tabelle tramite malloc
*punt_v_regioni=( t_regione* )malloc( n_r*sizeof( t_regione ) );
*punt_v_candidati=( t_candidato* )malloc( n_c*sizeof( t_candidato ) );
v_r=*punt_v_regioni;
v_c=*punt_v_candidati;

// [3] Immissione dei dati delle regioni
while( fscanf( f_archivio, "%s %s", Regione, NomeFile )>0 ) {
        strcpy( v_r[indice_r].Regione, Regione );
        if( !( f_regione=fopen( NomeFile, “rb” )  )  ) {
                printf( "Errore in apertura file %s\n", NomeFile );
                fclose( f_archivio );
                return -1;
        }
        v_r[indice_r].N_Cand = 0;
        while( fread( &temp, sizeof( t_cand_in ), 1, f_regione ) ) {
                v_r[indice_r].N_Cand++;
                v_c[indice_c].Codice=temp.Codice;
                strcpy(v_c[indice_c].Cognome, temp.Cognome );
                strcpy(v_c[indice_c].Telefono, temp.Telefono );
                v_c[indice_c].Data=temp.Data;
                v_c[indice_c].Regione=indice_r;
                indice_c++;
        }
        fclose( f_regione );
        indice_r++;
} // ( fine while )
fclose( f_archivio );

// [4] Immissione dei dati delle quote
if( !(f_quota=fopen( nome_file_quota, "r" ) ) ) {
        printf( "Errore in apertura file %s\n", nome_file_quota );
        return -1;
}
```

```
        while( fscanf( f_quota, "%d %f", &Codice, &Quota )>0 ) {
                indice_c=indice( Codice, v_c, n_c ); // funzione ausiliaria
                if( indice_c<n_c ) {
                        v_c[indice_c].Quota=Quota;
                        v_r[v_c[indice_c].Regione].Tot_Q+=Quota;
                }
                else {
                        printf( "Candidato non trovato: codice %d\n", Codice );
                        // non lo considero un errore
                }
        }
        fclose( f_quota );

        // [5] Restituzione risultato (0 in caso di successo)
        return 0;
}
```

Implementazione delle funzioni ausiliarie

```
int indice( int Codice, t_candidato *v, int dim ) {
        int i;
        for( i=0; i<dim; i++ )
                if( v[i].Codice==Codice )
                        break;
        return i;
}
```