Eserciziario

Archivi e file

Esercizio – File binari

1) Crea un file binario (STUDENTI) con la seguente struttura (tracciato record):
   Matricola – integer - chiave primaria
   Cognome – stringa
   Nome - stringa
   Sesso – char (m/f)
   DataNascita – record di 3 campi di tipo integer (GGMMAAAA)
   Telefono – stringa
   Indirizzo - stringa
   Titolo di studio – stringa (professionale/tecnico/liceo/laurea)
   Corso di Laurea – stringa (informatica, fisica, matematica, ...)
   Cancellato: char (0 --> no, 1 --> si)

2) Inserisci alcuni record (inserisci a mano matricole diverse)
Menù
a) Stampa il file
b) Ricerca per matricola (parametro, chiave primaria)
c) Ricerca per titolo di studio (parametro, chiave secondaria)

Note:
Apri e chiudi il file in ciascuna funzione

Soluzione (creazione archivio e inserimento dati)

```
struct data{
	int giorno;
	int mese;
	int anno;
};

struct studente{
	int matricola;			//chiave primaria
	char cognome[23];
	char nome[21];
	char sesso;			//'m', 'f'
	data nascita;
	char telefono[11];
	char indirizzo[31];
	char titoloStudio[19];
	char corsoLaurea[21];
	char cancellato;		//'0' --> no, '1' --> si
};
```

Soluzione

```
int main()
{
studente stud;
int numStud;
int count = 0;
FILE *fp;

    fp=fopen("studenti.dat","wb");

    printf("Quanti studenti vuoi inserire?\n");
    scanf("%d", &numStud);
    fflush(stdin);

    while(count < numStud)
    {
        printf("Digita maricola\n");
        scanf("%d",&stud.matricola);
        fflush(stdin);

        printf("Digita cognome\n");
        gets(stud.cognome);

        printf("Digita nome\n");
        gets(stud.nome);

        printf("Digita sesso (m/f)\n");
        scanf("%c",&stud.sesso);
        fflush(stdin);
```

Soluzione

```
        printf("Digita la data di nascita\n");
        printf("Digita il giorno\n");
        scanf("%d",&stud.nascita.giorno);
        fflush(stdin);
        printf("Digita il mese\n");
        scanf("%d",&stud.nascita.mese);
        fflush(stdin);
        printf("Digita l'anno\n");
        scanf("%d",&stud.nascita.anno);
        fflush(stdin);

        printf("Digita telefono\n");
        gets(stud.telefono);

        printf("Digita indirizzo\n");
        gets(stud.indirizzo);

        printf("Digita titolo studio\n");
        gets(stud.titoloStudio);

        printf("Digita corso laurea\n");
        gets(stud.corsoLaurea);

        stud.cancellato = '0';

        fwrite(&stud,sizeof(studente),1,fp);
        count++;
    }

    fclose(fp);

}
```

Soluzione (gestione archivio: stampa)

```
int main()
{
studente stud;
int numStud;
int count = 0;
FILE *fp;

    fp=fopen("studenti.dat","rb");


    fread(&stud,sizeof(studente),1,fp);

    while(!feof(fp))
    {

        printf("%d\n",stud.matricola);
        printf("%s\n",stud.cognome);
        printf("%s\n",stud.nome);
        printf("%c\n",stud.sesso);
        printf("%d\n",stud.nascita.giorno);
        printf("%d\n",stud.nascita.mese);
        printf("%d\n",stud.nascita.anno);
        printf("%s\n",stud.telefono);
        printf("%s\n",stud.indirizzo);
        printf("%s\n",stud.titoloStudio);
        printf("%s\n",stud.corsoLaurea);

        fread(&stud,sizeof(studente),1,fp);
    }

    fclose(fp);
```

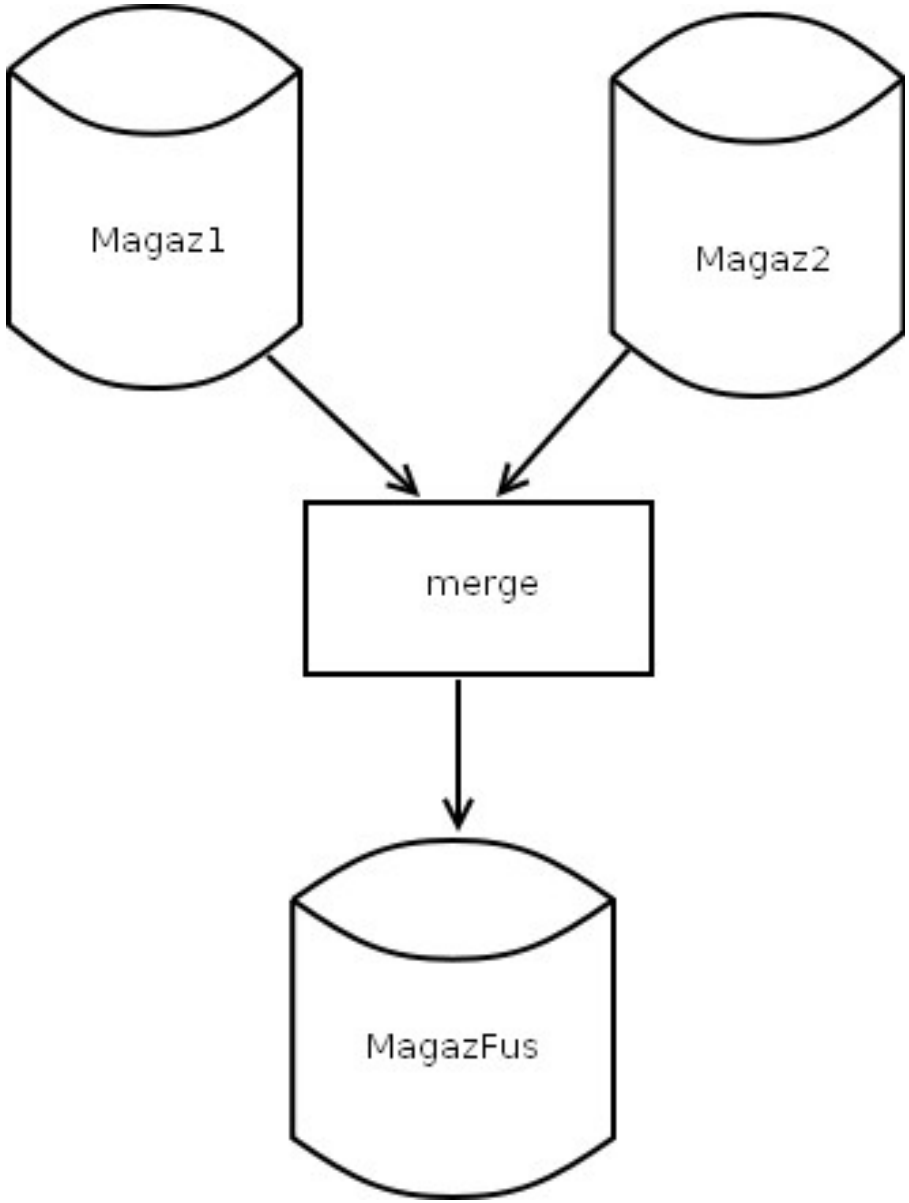
Esercizio – File binari

- Scrivi un pgm C che esegua la fusione (merge) di due file binari ordinati.

- Tracciato record:
  - codArt: stringa //chiave primaria
  - quantità: intero

- Vediamo per primo la versione in memoria centrale. Il vettore contiene solo la chiave (numerica).

DFD



Soluzione in memoria centrale

```
#define DIM1 10
#define DIM2 20
#define DIM_FUS  DIM1 + DIM2

int Fusione(const int vett1[], int numElem1, const int vett2[], int numElem2,
        int vettFus[]);

int main(void)
{
int vett1[]={1,2,3,4,6,8};
int vett2[]={2,5,6,7,8,9,10,12};
int vettFus[DIM_FUS];
int numElem1=sizeof(vett1)/sizeof(int);
int numElem2=sizeof(vett2)/sizeof(int);
int numElem3;
int i;

      numElem3=Fusione(vett1,numElem1,vett2,numElem2,vettFus);

      printf("Num. elem. nel vettore fusione %d\n",numElem3);

      for(i=0; i<numElem3; i++)
      printf("%d\n",vettFus[i]);

      getchar();
      return 0;
}
```

Soluzione

```
int Fusione(const int vett1[], int numElem1, const int vett2[], int numElem2, int vettFus[])
{
int count1=0;
int count2=0;
int count3=0;

  while((count1<numElem1) && (count2<numElem2))
  {
    if (vett1[count1] < vett2[count2])
    {
                vettFus[count3++]=vett1[count1++];
    }
    else if(vett1[count1] > vett2[count2])
        {
          vettFus[count3++]=vett2[count2++];
        }
    else
        {
          vettFus[count3++]=vett1[count1++];
          count2++;
        }
  }
```

Soluzione

```
 for(; count1 < numElem1; count1++,count3++)
      vettFus[count3]=vett1[count1];

  for(; count2 < numElem2; count2++,count3++)
      vettFus[count3]=vett2[count2];

  return count3;
}
```

Soluzione in memoria di massa

```
struct prodotto{
    char codProd[6];
    unsigned int quantita;
};

int main()
{
FILE *fpMag1;
FILE *fpMag2;
FILE *fpFus;
struct prodotto art1,art2,fus;

      fpMag1=fopen("magaz1.dat","rb");
      fpMag2=fopen("magaz2.dat","rb");
      fpFus=fopen("fusione.dat","wb");
```

Soluzione

```
fread(&art1,sizeof(art1),1,fpMag1);
fread(&art2,sizeof(art2),1,fpMag2);
while( !feof(fpMag1) && !feof(fpMag2) )
{
        if(strcmp(art1.codProd,art2.codProd)==0)
        {
                        strcpy(fus.codProd,art1.codProd);
                        fus.quantita=art1.quantita+art2.quantita;
                        fwrite(&fus,sizeof(fus),1,fpFus);
                        fread(&art1,sizeof(art1),1,fpMag1);
                        fread(&art2,sizeof(art2),1,fpMag2);
        }
        else if(strcmp(art1.codProd,art2.codProd)>0)
        {
                    strcpy(fus.codProd,art2.codProd);
                    fus.quantita=art2.quantita;
                    fwrite(&fus,sizeof(fus),1,fpFus);
                    fread(&art2,sizeof(art2),1,fpMag2);
        }
        else
        {
                    strcpy(fus.codProd,art1.codProd);
                    fus.quantita=art1.quantita;
                    fwrite(&fus,sizeof(fus),1,fpFus);
                    fread(&art1,sizeof(art1),1,fpMag1);
        }
}
```

Soluzione

```
while(!feof(fpMag1))
{
  strcpy(fus.codProd,art1.codProd);
  fus.quantita=art1.quantita;
  fwrite(&fus,sizeof(fus),1,fpFus);
  fread(&art1,sizeof(art1),1,fpMag1);

}

while(!feof(fpMag2))
{
  strcpy(fus.codProd,art2.codProd);
  fus.quantita=art2.quantita;
  fwrite(&fus,sizeof(fus),1,fpFus);
  fread(&art2,sizeof(art2),1,fpMag2);
}

fclose(fpMag1);
fclose(fpMag2);
fclose(fpFus);

getchar();
return 0;
}
```

Elaborazione per rottura di codice o di livello

- E’ un tipico esempio di elaborazione sequenziale di un archivio con organizzazione sequenziale.
- Problema:

  Dato un file sequenziale nel quale sono memorizzate le vendite effettuate da diversi rappresentanti, si realizzi un pgm per calcolare e stampare l’ammontare complessivo del venduto distinto per regione, zona di attività e rappresentante.
- Il layout della stampa può essere il seguente:

  elaborazione rottura di codice o livello.pdf

Elaborazione per rottura di codice o di livello

Ipotesi: si suppone che il file su cui si opera sia ordinato gerarchicamente nel seguente modo:

- in sequenza troviamo i record che si riferiscono ad una stessa regione
- all'interno di ciascuna regione saranno contigui tutti i record che riferiscono ad una stessa zona
- infine, in ciascuna zona, saranno contigui tutti i record di ciascun rappresentante.

Tracciato record:

FATTURATO (codReg, codZona, codRappr, venduto)

Elaborazione per rottura di codice o di livello

Note:

- Codici ….. e i nomi per esteso?
- Elaborazione batch o elaborazione interattiva ?
- Algoritmi di sort interni/esterni
- Archivio = insieme di registrazioni (record) omogenee (= dello stesso tipo)
- Archivio: concetti di organizzazione e metodo d'accesso
- Dispositivi di memoria di massa: nastri, dischi, …
- Concetto di elaborazione di massa
- Record a lunghezza fissa/variabile
- Diagramma di flusso dati

Esercizio – File binari

La società STAT effettua elaborazioni statistiche. Recentemente il centro di calcolo della STAT ha elaborato una tabella statistica dei comuni italiani così organizzata:

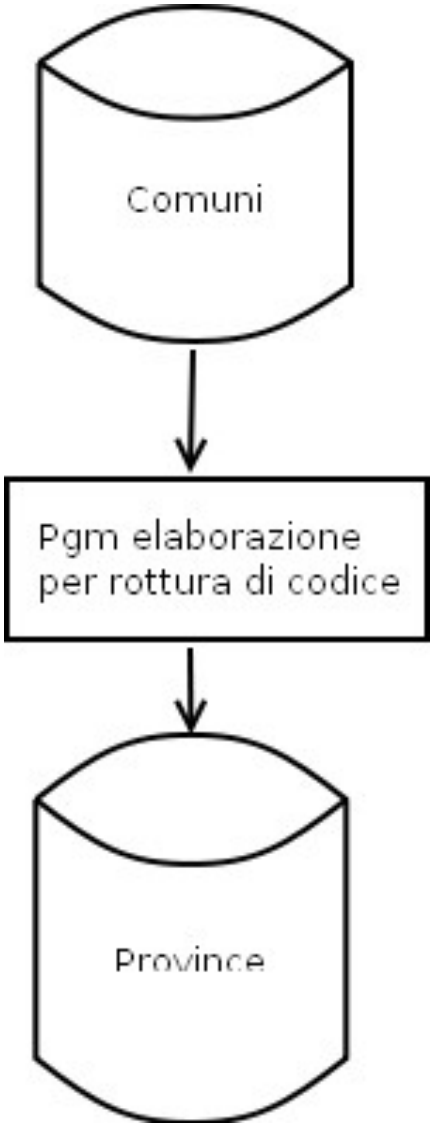
* la tabella (file binario) contiene l'elenco di tutti i comuni italiani;
* per ogni comune, la tabella contiene:
    1. il nome del comune;
    2. la sigla della provincia di appartenenza;
    3. il numero di abitanti del comune;
* la tabella è ordinata per ordine crescente di sigla delle province e, nell'ambito della stessa provincia, in base al nome del comune.

Dalla tabella statistica dei comuni la STAT vuole ottenere una tabella (file binario), detta tabella statistica delle province, così organizzata:

* la tabella deve contenere l'elenco ordinato di tutte le province italiane;
* per ogni provincia, la tabella deve contenere:
    1. la sigla della provincia;
    2. il numero dei comuni che ne fanno parte;
    3. il numero complessivo di abitanti della provincia.

Dopo aver dichiarato le strutture dati, scrivi un pgm C che realizzi quanto sopra descritto, cioè che a partire dalla tabella COMUNI crei la tabella PROVINCE.

DFD



Soluzione

```
struct comune{
   char nome[31];
   char prov[3];
   unsigned int numAbitanti;
};
struct provincia{
   char sigla[3];
   unsigned short int numComuni;
   unsigned int numAbitanti;
};
int main(void)
{
FILE * fpComuni;
FILE * fpProvincie;
comune comun;
provincia prov;
unsigned short int totComuni;
unsigned int totAbitanti;
char ultProv[3];
```

Soluzione

Esercizio – File binari

- Scrivi due funzioni che contano il numero di record di un archivio: la prima sfruttando solo l'accesso sequenziale, l'altra quello diretto.

```
int main()
{
    printf("Accesso sequenziale:  %d\n",contaSequenziale());
    printf("Accesso diretto:  %d\n",contaDiretto());

    getchar();
    return 0;
}
```

Soluzione

```
int contaSequenziale()
{
FILE *fp;
studente stud;
int count=0;

        fp=fopen("studenti.dat","rb");

        while(fread(&stud,sizeof(studente),1,fp))
        {
           count++;
        }

        fclose(fp);
        return count;
}
```

```
int contaDiretto()
{
FILE *fp;
long dim;

     fp=fopen("studenti.dat","rb");

     fseek(fp,0L,SEEK_END);
     dim=ftell(fp);
     fclose(fp);
     return dim/sizeof(studente);
}
```

Esercizio – File binari

- Inserire nel menù dell'esercizio n.1 le seguenti voci:
  - Aggiornamento dei campi telefono e/o indirizzo, data la matricola
  - Cancellazione (logica) di uno studente, data la matricola
  - Inserimento di un nuovo studente, la matricola (chiave primaria) viene assegnata automaticamente dal sistema (progressivo)
  - Cancellazione fisica

Esercizio – File binari

Dato un file binario di interi “interi.dat”, scrivi un pgm C che lo modifichi incrementando di 1 gli elementi positivi che precedono il valore 0, sia con:

- accesso sequenziale, utilizzando un file temporaneo;

sia con:

- accesso diretto, utilizzando la funzione di libreria fseek().

Esempio:

File “interi.dat” prima

1	0	0	6	8	0	9

File “interi.dat” modificato

2	0	0	6	9	0	9

NB Scrivi due pgm C, uno che utilizza l’accesso sequenziale e l’altro l’accesso diretto.

Soluzione – versione con accesso sequenziale

Soluzione – versione con accesso diretto