Fondamenti di Informatica e Laboratorio T-AB
Ingegneria Elettronica e Telecomunicazioni e
Ingegneria dell'Automazione
a.a. 2009/2010

Lab 16
Gestione file binari

Esercizio 1

- Un registratore di cassa registra su di un file binario alcuni dati relativi agli scontrini emessi. In particolare, tramite una struttura dati di nome scontrino, il registratore di cassa tiene traccia dell'importo (un float) e del numero di oggetti acquistati (un intero).
- In un apposito file registratore.h, si definisca una struttura dati "scontrino" atta a contenere/rappresentare i dati forniti dal registratore di cassa.
- In un apposito modulo software registratore.h / registratore.c, si devono realizzare due funzioni, una di lettura ed una di scrittura delle strutture dati scontrino:

```
int leggi(FILE * fp, scontrino * dest);
int scrivi(FILE * fp, scontrino src);
```

Esercizio 1

- Si realizzi un programma main che apra il file binario reg.dat in scrittura, e poi simuli il funzionamento del registratore di cassa per testare il modulo registratore. In particolare il programma chieda all'utente di inserire coppie importo / numero di oggetti, e le registri sul file binario usando la funzione scrivi(…). L'utente può segnalare la fine dell'inserimento dei valori indicando una coppia 0.00/0, che ovviamente non deve essere registrata nel file.
- Terminata la fase di inserimento, il programma chiuda il file e lo ri-apra in lettura, e stampi a video tutte le coppie memorizzate (per la lettura, si utilizzi la funzione leggi(…))
- Al termine il main chiuda correttamente il file.

Esercizio 1 – Soluzione

- “registratore.h”:

```
#include <stdio.h>

typedef struct {
      float val;
      int num;
} scontrino;


int leggi(FILE * fp, scontrino * dest);
int scrivi(FILE * fp, scontrino src);
```

Esercizio 1 – Soluzione

- “registratore.c”:

```
#include "registratore.h"


int leggi(FILE * fp, scontrino * dest) {
      return fread(dest, sizeof(scontrino), 1, fp);
}

int scrivi(FILE * fp, scontrino src) {
      return fwrite(&src, sizeof(scontrino), 1, fp);
}
```

Esercizio 1 – Soluzione

- “main.c”:

```
#include <stdio.h>
#include <stdlib.h>
#include "registratore.h"

int main(void) {
        scontrino temp;
        FILE * fp;

        if ( (fp=fopen("reg.dat", "wb")) == NULL)
                exit(-1);
        do {
                printf("Inserisci valore e numero di pezzi: ");
                scanf("%f %d", &(temp.val), &(temp.num) );
                if (temp.val!=0 || temp.num!=0)scrivi(fp, temp);
        } while (temp.val!=0 || temp.num!=0);
        fclose(fp);

        if ( (fp=fopen("reg.dat", "rb")) == NULL)
                exit(-1);
        while (leggi(fp, &temp)>0)
                printf("Prezzo: %f, pezzi: %d\n", temp.val, temp.num);
        fclose(fp);
        system("PAUSE"); return (0); }
```

Esercizio 2

- Un sito web per le speculazioni borsistiche registra su un file di testo gli andamenti di alcuni titoli azionari. In particolare, su ogni riga del file, registra il nome di un titolo (al più 63 caratteri senza spazi), i valori di apertura e di chiusura del titolo (tramite due float), ed il giorno dell'anno corrente in cui è stato monitorato il titolo (un intero).
- Nel file non c'è nessun ordine preciso con cui le righe sono state memorizzate, e mano a mano che passano i giorni vengono registrate più e più righe relative allo stesso titolo. Il sito web ha comunque l'accortezza di registrare nel file al massimo 100 righe relative allo stesso titolo.
- Si vuole realizzare un programma per valutare la volatilità di un titolo (cioè i valori minimi e massimi raggiunti), relativamente alle informazioni presenti sul file.

Esercizio 2

A tal scopo, in un opportuno modulo software azioni.h/azioni.c:

- Si definisca una apposita struttura dati, di nome “azione”, per memorizzare le informazioni relative ad un titolo.
- Si realizzi una funzione:

```
int leggi(FILE* fp, azione dest[], int dim, char * nome);
```

- che, ricevuti in ingresso un puntatore ad un file già opportunamente aperto, un array di strutture azione e la sua dimensione fisica dim, ed il nome di un titolo azionario, provveda a copiare in dest tutte le strutture relative all’azione specificata. La funzione deve restituire il numero di elementi copiati, ovvero la dimensione logica del vettore dest.

Esercizio 2

Quindi, in un altro modulo software “trova.h/trova.c”:

- Si realizzi una funzione:

```
azione trovaMin(azione src[], int dim, float* val);
```

  che, ricevuti in ingresso un array di strutture azione e la sua dimensione dim, ed il nome di una azione, determini il valore minimo raggiunto dal titolo azionario (le cui fluttuazioni sono memorizzate in src). La funzione deve restituire la struttura dati azione relativa a quando si è verificato il minimo, e tramite il parametro val passato per riferimento deve restituire il minimo raggiunto.

- Si realizzi una funzione:

```
azione trovaMax(azione src[], int dim, float* val);
```

  che, ricevuti in ingresso un array di strutture azione e la sua dimensione dim, ed il nome di una azione, determini il valore massimo raggiunto dal titolo azionario (le cui fluttuazioni sono memorizzate in src). La funzione deve restituire la struttura dati azione relativa a quando si è verificato il massimo, e tramite il parametro val passato per riferimento deve restituire il massimo raggiunto.

Esercizio 2

- Si realizzi un programma main.c che, utilizzando le funzioni già definite, provveda a chiedere all'utente il nome di un titolo azionario, e stampi a video i valori minimi e massimi raggiunti dalle quotazioni del titolo.
- Si estenda il modulo "trova.h/trova.c" con una funzione opportuna per calcolare la media del valore del titolo azionario rispetto alle varie quotazioni registrate, e poi si provveda a stampare a video l'oscillazione (in percentuale rispetto al valore medio) del minimo e del massimo raggiunti dal titolo in borsa.

Esercizio 2 – Soluzione

- “azioni.h”:

```
#include <stdio.h>
#include <string.h>


typedef struct {
        char nome[64];
        float ap;
        float cl;
        int day;
} azione;


int leggi(FILE* fp, azione dest[], int dim, char * nome);
```

Esercizio 2 – Soluzione

- “azioni.c”:

```
#include "azioni.h"


int leggi(FILE* fp, azione dest[], int dim, char * nome) {
        azione temp;
        int result = 0;

        while (fscanf(fp, "%s %f %f %d", temp.nome, &(temp.ap),
                                &(temp.cl), &(temp.day)) == 4 && result<dim) {
                if (strcmp(temp.nome, nome) == 0) {
                        dest[result] = temp;
                        result++;
                }
        }
        return result;
}
```

Esercizio 2 – Soluzione

- “trova.h”:

```
#include "azioni.h"

azione trovaMin(azione src[], int dim, float* val);
azione trovaMax(azione src[], int dim, float* val);
float media(azione src[], int dim);
```

Esercizio 2 – Soluzione

- “trova.c”:

```
#include "trova.h"

azione trovaMin(azione src[], int dim, float* val) {
        int i;
        azione result;

        result = src[0]; // ATTENZIONE!!! SE dim == 0?
        if (result.ap < result.cl)
                *val = result.ap;
        else
                *val = result.cl;
        for (i=1; i<dim; i++) {
                if (src[i].ap < *val) {
                        *val = src[i].ap;
                        result = src[i];
                }
                if (src[i].cl < *val) {
                        *val = src[i].cl;
                        result = src[i];
                }
        }
        return result;
}
```

Esercizio 2 – Soluzione

- “trova.c”:

```
azione trovaMax(azione src[], int dim, float* val) {
        int i;
        azione result;

        *val = -1;
        result = src[0]; // ATTENZIONE!!! SE dim == 0?

        for (i=0; i<dim; i++) {
                if (src[i].ap > *val) {
                        *val = src[i].ap;
                        result = src[i];
                }
                if (src[i].cl > *val) {
                        *val = src[i].cl;
                        result = src[i];
                }
        }
        return result;
}
```

Esercizio 2 – Soluzione

- “trova.c”:

```
float media(azione src[], int dim) {
        int tot=0;
        int i;
        float sum=0;

        for (i=0; i<dim; i++) {
                sum = sum + src[i].ap + src[i].cl;
                tot+=2;
        }
        return sum/tot;
}
```

Esercizio 2 – Soluzione

- "main.c":

```
#include <stdio.h>
#include <stdlib.h>
// #include "azioni.h" NON NECESSARIO perche' gia' incluso da trova.h...
#include "trova.h"

int main(void) {
        FILE * fp;
        azione temp1, temp2;
        azione lista[100];
        int dim;
        float min, max, med;
        char nome[64];

        if ((fp=fopen("azioni.txt", "r")) == NULL)
                exit(-1);
        printf("Nome titolo azionario: ");
        scanf("%s", nome);
        dim = leggi(fp, lista, 100, nome);
        temp1 = trovaMin(lista, dim, &min);
        temp2 = trovaMax(lista, dim, &max);

        ...
```

Esercizio 2 – Soluzione

- “main.c”:

```
        ...

        printf("Minimo raggiunto da %s il giorno %d, valore %f euro.\n",
                temp1.nome, temp1.day, min);
        printf("Massimo raggiunto da %s il giorno %d, valore %f euro.\n",
                temp2.nome, temp2.day, max);
        med = media(lista, dim);
        printf("Media: %f\n", med);
        printf("Volatilita': %f...%f\n", (min-med)/med*100, (max-med)/med*100);


        system("PAUSE");

        return (0);
}
```

Esercizio 3

Un venditore di ortofrutta è solito fare credito ai propri clienti. Al fine di tenere traccia dei crediti, utilizza una funzione del registratore di cassa che salva, su un file binario di nome "log.dat", strutture dati del tipo **`transaction`** contenenti i seguenti dati:

- una stringa **`customer`**, contenente il nome del cliente (al più 128 caratteri, senza spazi);
- un intero **`transactionId`**, recante un codice identificativo della transazione;
- un float **`value`**, contenente l'ammontare del credito concesso.

```
#define DIM 129
typedef struct {
        char customer[DIM];
        int transactionId;
        float value;
} transaction;
```

Esercizio 3

Al momento di riscuotere i crediti, il commerciante deve però poter accedere ai valori registrati nel file binario.

A tal scopo, egli vuole avere un programma che, richiesto il nome del cliente, scriva su un file in formato testo gli importi relativi solo al cliente specificato.

Inoltre il file deve avere come nome il nome del cliente con l'aggiunta dell'estensione "`.txt`". Ad esempio, se il cliente richiesto si chiama "Federico", allora il file dovrà chiamarsi "`Federico.txt`".

Esercizio 3

In un apposito modulo software, denominato "registratore.h / registratore.c", dopo aver definito opportunamente le strutture dati necessarie, si realizzi una procedura

```
void copy(FILE* source, FILE* dest, char *name, int *result)
```

che, ricevuti in ingresso un puntatore `source` al file binario, un puntatore `dest` al file di testo, un puntatore a carattere `name` e un intero `result` passato per riferimento, copi su `dest` tutti gli importi presenti in `source` e relativi al cliente specificato con parametro `name`

La funzione deve tenere traccia del numero di importi di credito copiati e deve restituire tale numero tramite il parametro `result`. Gli importi devono essere scritti su una sola linea, separati da uno spazio, con al termine un carattere di "newline" ('`\n`'). Al fine di confrontare due stringhe, si utilizzi la funzione **`strcmp(…)`**, che restituisce 0 se le due stringhe passate come parametri sono identiche

Esercizio 3

Realizzare poi un programma "main.c" che chieda inizialmente all'utente il nome di un cliente. Per creare un opportuno nome per il file di destinazione, il candidato può utilizzare le funzioni di libreria:

- `strcpy(char *s, char *ct)`, che provvede a copiare la stringa ct nella stringa s;
- `strcat(char * s, char * ct)`, che concatena il contenuto della stringa ct in fondo alla stringa s (si faccia particolare attenzione a dimensionare opportunamente la stringa s per contenere i 4 caratteri dell'estensione ".txt"). La stringa s, al termine dell'invocazione, è sempre una stringa ben formata

Dopo aver aperto i file nell'opportuna modalità di lettura/scrittura, il programma utilizzi la funzione copy(…) definita al punto precedente per filtrare i dati. Il programma stampi infine a video il numero totale di crediti che sono stati copiati da un file all'altro

Esercizio 3 – Soluzione

```
void copy(FILE *source, FILE *dest, char *name, int
*result) {

      transaction temp;
      *result = 0;

      while (
        fread(&temp, sizeof(transaction), 1, source) >0
      ) {
            if (strcmp(name, temp.customer) == 0) {
                  fprintf(dest, "%f ", temp.value);
                  (*result)++;
            }
      }
      fprintf(dest, "\n");
}
```

Esercizio 3 – Soluzione

```
int main() {
        char name[DIM], filename[DIM+4];
        FILE *source, *dest;
        int result = 0;

        printf("Insert customer name: ");
        scanf("%s", name);
        if ((source = fopen("log.dat", "rb")) == NULL) {
                        printf("Error opening the file %s\n", "log.dat");
                        exit(-1);}

        strcpy(filename, name);
        strcat(filename, ".txt");
        if ((dest = fopen(filename, "w")) == NULL) {
                        printf ("Error opening the file %s\n", filename);
                        exit(-1);}

        copy(source, dest, name, &result);

        fclose(source); fclose(dest);
        printf("%d records copied by log.dat to %s\n",result,filename);
        return 0;
}
```