Fondamenti di Informatica e Laboratorio T-AB
Ingegneria Elettronica e Telecomunicazioni

Lab 9
Strutture e programmi su più file

Esercizio 1

- Sia data la struttura

```
struct time
{
  int hour, minute, second;
};
```

- Per semplicità si può definire il tipo Time

```
typedef struct time Time;
```

Esercizio 1

Si realizzi in un modulo **tempo.h/tempo.c** un insieme di funzioni per la gestione del tipo Time. In particolare:

- Si realizzi una funzione

  ```
  Time leggiTime()
  ```

  che legga da input ore, minuti e secondi, e restituisca una struttura di tipo Time opportunamente inizializzato coi valori letti
- Si realizzi una funzione

  ```
  int leggiMoreTimes(Time v[], int dim)
  ```

  La funzione deve leggere da input delle strutture Time (a tal scopo si utilizzi la funzione definita sopra) e salvarle nel vettore v, di dimensione fisica dim. La funzione deve restituire il numero di elementi letti. La lettura termina se l'utente inserisce un tempo con ora negativa.

Esercizio 1

- Si progetti una funzione in grado di calcolare la differenza fra due strutture **`Time`** e che restituisca il risultato in termini di una nuova struttura **`Time`**

- L'interfaccia della funzione è facilmente desumibile dalle specifiche:

```
Time subtract(Time t1, Time t2);
```

- Due possibili approcci:
  1. Trasformare in secondi, eseguire la differenza, trasformare in ore, minuti, secondi
  2. Eseguire la sottrazione direttamente tenendo conto dei riporti

Esercizio 1 – Soluzione

File "Tempo.h"

```
#include <stdio.h>

#ifndef TEMPO_H
#define TEMPO_H

struct time {
     int hour, minute, second;
};
typedef struct time Time;

#endif

Time leggiTime();
int leggiMoreTimes(Time v[], int dim);
Time subtract1(Time t1, Time t2);
Time subtract2(Time t1, Time t2);
```

Esercizio 1 – Soluzione

File "Tempo.c"

```
#include "tempo.h"


Time leggiTime() {
     Time result;

     printf("Ore: ");
     scanf("%d", &result.hour );
     printf("Minuti: ");
     scanf("%d", &result.minute );
     printf("Secondi: ");
     scanf("%d", &result.second );

     return result;
}
```

…

Esercizio 1 – Soluzione

File "Tempo.c"

```
…
int leggiMoreTimes(Time v[], int dim) {
     int result;
     Time temp;

     result = 0;
     do {
        temp = leggiTime();
        if (temp.hour>=0 && result<dim) {
               v[result] = temp;
               result++;
        }
     } while (temp.hour>=0 && result<dim);
     return result;
}
…
```

Esercizio 1 – Soluzione

File "Tempo.c"

…

```
Time subtract1(Time t1, Time t2)
{
    int s1, s2, sResult;
    Time result;

    s1 = t1.hour * 3600 + t1.minute * 60 + t1.second;
    s2 = t2.hour * 3600 + t2.minute * 60 + t2.second;
    sResult = s1 - s2;

    result.hour = sResult / 3600;
    sResult = sResult % 3600;
    result.minute = sResult / 60;
    sResult = sResult % 60;
    result.second = sResult;

    return result;
}
```

Esercizio 1 – Soluzione

File "Tempo.c"

```
…
Time subtract2(Time t1, Time t2) {
    Time result;
    int carry;
    result.second = t1.second - t2.second;
    carry = 0;
    if (result.second < 0) {
        result.second = 60 + result.second;
        carry = -1;
    }
    result.minute = t1.minute - t2.minute + carry;
    carry = 0;
    if (result.minute < 0) {
        result.minute = 60 + result.minute;
        carry = -1;
    }
    result.hour = t1.hour - t2.hour + carry;
    return result;
}
```

Esercizio 2

Una compagnia di autobus che effettua servizio su lunghe distanze vuole realizzare un programma di controllo delle prenotazioni dei posti.

A tal scopo rappresenta ogni prenotazione tramite una struttura **`booking`** contenente nome del cliente (al massimo 1023 caratteri, senza spazi) e numero del posto prenotato (un intero).

Le prenotazioni effettuate vengono registrate tramite un array (di dimensione prefissata **`DIM`**) di strutture **`booking`**, di dimensione logica iniziale pari a 0.

Si realizzi il modulo C gestione.h/gestione.c, contenente la struttura dati booking e le seguenti funzioni…

Esercizio 2

a) Si realizzi una funzione:

```
int leggi(booking * dest);
```

La funzione legge da input una struttura di tipo booking, e provvede a memorizzarla in dest. La funzione deve restituire 1 se è stata letta una nuova prenotazione, 0 altrimenti (cioè nel caso in cui il nome del passeggero è “fine”).

Esercizio 2

b) Si realizzi una funzione:

```
int assegna( booking list[],
             int dim,
             int * lengthList,
             char * name,
             int pref)
```

La funzione riceve in ingresso l'array di prenotazioni e la sua dimensione fisica e logica, e poi il nome del cliente ed il posto da lui indicato. La funzione deve controllare che il posto indicato non sia già stato assegnato, ed in caso contrario deve restituire il valore 1.

Esercizio 2

Qualora invece il posto sia ancora libero, la funzione deve assegnare tale posto al cliente copiando i dati della prenotazione nell'ultima posizione libera nell'array, e deve provvedere ad aggiornare correttamente la dimensione logica dell'array. In questo secondo caso la funzione deve invece restituire come valore uno 0, indicante il successo nella prenotazione.

Al fine di copiare il nome del cliente, si utilizzi la funzione di libreria

```
char * strcpy(char * s, char * ct)
```

che copia ct in s (terminatore compreso).

Esercizio 2

c) Si realizzi un programma main (file main.c) che chieda all'operatore il nome di un utente, e di seguito il posto prescelto (a tal fine si usi la funzione di cui al punto a)). Il programma deve cercare di registrare la prenotazione tramite la funzione **`assegna`**; qualora l'operazione di prenotazione fallisca (perché il posto risulta essere già assegnato), il programma provveda a chiedere all'operatore un nuovo posto, finché non si riesca ad effettuare la prenotazione.

Esercizio 2

Qualora l'operatore inserisca il nome "fine", il programma deve terminare; qualora invece venga inserita la stringa "stampa", il programma deve stampare a video le prenotazioni già effettuate.

A tal scopo si usi la funzione di libreria:

```
int strcmp(char * ct, char * cs)
```

che restituisce 0 se e solo se le due stringhe sono identiche (lessicograficamente).

Esercizio 2 – Soluzione

File “gestione.h”:

```
#include <stdio.h>
#include <string.h>
#define MAX 1024
#define DIM 10

#ifndef GESTIONE_H
#define GESTIONE_H

typedef struct {
    char name[MAX];
    int seat;
} booking;

#endif

int leggi(booking * dest);
int assegna( booking list[], int * lengthList, int dim, char * name, int pref);
int assegna2( booking list[], int * lengthList, int dim, booking temp);
int stampaBooking(booking list[], int lengthList);
```

Esercizio 2 – Soluzione

File "gestione.c":

```
#include "gestione.h"

int leggi(booking * dest) {
      printf("Inserire il nome: ");
      scanf("%s", (*dest).name );

      if (strcmp("fine", dest->name)==0 ||
                  strcmp("stampa", dest->name)==0)
          return 0;
      else {
          printf("Posto preferito: ");
          scanf("%d", &(dest->seat));
          return 1;
      }
}
...
```

Esercizio 2 – Soluzione

```
int assegna( booking list[], int * lengthList, int dim, char * name, int pref) {
     int i=0;
     int trovato = 0;

     while (( i < *lengthList) && !trovato) {
         if (list[i].seat == pref)
                  trovato = 1;
         i++;
     }
     if (!trovato && *lengthList<dim) {
         list[*lengthList].seat = pref;
         strcpy(list[*lengthList].name, name);
         (*lengthList)++;
         return 0;
     }
     else return 1;
}
```

Esercizio 2 – Soluzione

```
int assegna2( booking list[], int * lengthList, int dim, booking temp) {
     int i=0;
     int trovato = 0;

     while ( (i < *lengthList) && !trovato) {
         if ( list[i].seat == temp.seat  )
                  trovato = 1;
         i++;
     }
     if (!trovato && *lengthList<dim) {
         list[*lengthList] = temp;
         (*lengthList)++;
         return 0;
     }
     else
         return 1;
}
```

Esercizio 2 – Soluzione

```
int stampaBooking(booking list[], int lengthList) {
     int i=0;

     for (i=0; i<lengthList; i++)
        printf("%s: %d\n", list[i].name, list[i].seat);

     return 0;
}
```

Esercizio 2 – Soluzione

File "main.c":

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

int main(void) {
     booking list[DIM];
     int lengthList = 0;
     booking temp;

…
```

Esercizio 2 – Soluzione

```
...
    do {
        if (leggi(&temp)) {
                if (assegna2(list, &lengthList, DIM, temp))
                        printf("Posto gia' occupato, pren. non effettuata!\n");
                else
                        printf("Prenotazione efffettuata con successo!\n");
        }
        else {
                if (strcmp("stampa", temp.name) == 0)
                        stampaBooking(list, lengthList);
        }
    } while (strcmp("fine", temp.name) != 0 && lengthList<DIM);

     return (0);
}
```