Laboratorio di Programmazione: Linguaggio C
Lezione 15 del 10 marzo 2014

Damiano Macedonio

Esercizio 1

Scrivere un programma C che richiede all'utente di inserire due date e calcola il numero di giorni trascorsi tra di esse. Il programma deve comprendere le seguenti funzioni.

`int main(void)`
: richiama opportunamente le funzioni seguenti.

`struct date insertDate()`
: Richiede all'utente di inserire una data e restituisce la corrispondente struttura.
Se la data inserita non è valida viene richiesta di nuovo.
Suggerimento: Usate la funzione `scanf` con stringa di formato `"%i/%i/%i"` per permettere all'utente di inserire una data nella forma: `12/3/2013`.

`bool checkDate(struct date d)`
: verifica se la data `d` è valida.
In dettaglio, tale funzione deve verificare che:

- L'anno sia composto da 4 cifre.
- Il mese sia compreso tra 1 e 12.
- Il numero di giorni sia compreso tra 1 e *max*, dove *max* è il numero massimo di giorni per il mese inserito.
  - Usare una funzione aggiuntiva che dato il mese e l'anno calcola il numero massimo di giorni per quel mese.
  - Un anno è bisestile se il suo numero è divisibile per 4 e non è divisibile per 100, oppure è divisibile per 400.

    ```
    year % 4 == 0 && (year % 100 != 0 || year % 400 == 0))
    ```

- La data complessiva sia maggiore o uguale a `1/3/1700`.

Suggerimento: Potete sfruttare `#include <stdbool.h>` per poter utilizzare il tipo `bool` che definisce i valori `true` e `false`.

`int elapsedDays(struct date d1, struct date d2)`
: Calcola i giorni trascorsi tra `d1` e `d2`.

Suggerimento: Il numero di giorni tra due date si piò calcolare applicando la seguente formula:

$$\texttt{elapsedDays(d1,d2)} = n(\texttt{d2}) - n(\texttt{d1})$$

con la funzione $n(\,)$ definita come segue

$$n(\texttt{d}) = \frac{1461 \cdot f(\texttt{d.year},\ \texttt{d.month})}{4} + \frac{153 \cdot g(\texttt{d.month})}{5} + \texttt{d.day} + k(\texttt{d})$$

dove

$$f(\texttt{d.year},\ \texttt{d.month}) = \begin{cases} \texttt{d.year} - 1 & \text{se } \texttt{d.month} \leq 2 \\ \texttt{d.year} & \text{altrimenti} \end{cases}$$

$$g(\texttt{d.month}) = \begin{cases} \texttt{d.month} + 13 & \text{se } \texttt{d.month} \leq 2 \\ \texttt{d.month} + 1 & \text{altrimenti} \end{cases}$$

$$k(\texttt{d}) = \begin{cases} 0 & \text{se d è seguente al 1/3/1900} \\ 1 & \text{se d è compresa tra 1/3/1800 e 28/2/1900} \\ 2 & \text{se d è compresa tra 1/3/1700 e 28/2/1800} \end{cases}$$

Esercizio 2

Scrivere un programma C che calcola il tempo trascorso (in ore, minuti e secondi) tra due istanti di tempo.

- Definire una struttura `time` che contiene ora, minuti e secondi.
- Il tempo trascorso tra `3:45:15` e `9:44:03` è 5 ore, 58 minuti e 48 secondi.

Il programma deve comprendere le funzioni:

`struct time insertTime()`
: Richiede all'utente di inserire un istante e restituisce la corrispondente struttura. Se l'istante inserito non è valido viene richiesto di nuovo.
Suggerimento: Usare la funzione `scanf` con stringa di formato `"%i:%i:%i"` per permettere all'utente di inserire una data nella forma: `12:03:33`

`bool checkTime(struct time)`
: Verifica se l'istante di tempo inserito è valido.
In particolare la funzione verifica che:
  - Il numero che rappresenta i secondi sia compreso tra 0 e 59.
  - Il numero che rappresenta i minuti sia compreso tra 0 e 59.
  - Il numero che rappresenta l'ora sia compreso tra 0 e 23.

`struct time elapsedTime(struct time t1, struct time t2)`
: Calcola il tempo trascorso tra `t1` e `t2`.

`int main(void)`
: Richiama opportunamente le altre funzioni.

Esercizio 3

Si consideri la regione $\mathcal{R}$ del piano cartesiano individuata dalle ascisse comprese tra 0 e 9 e dalle ordinate comprese tra 0 e 9.

Scrivere un programma C che chiede all'utente le coordinate di due punti `a` e `b` appartenenti a $\mathcal{R}$ e disegna il perimetro del rettangolo con i lati paralleli agli assi cartesiani che ha il segmento `ab` come diagonale. Le coordinate di `a` e `b` devono essere intere.

Il programma deve definire le seguenti strutture:

- Una struttura `punto` che contiene ascissa e ordinata.
- Un'opportuna struttura `rettangolo` che rappresenta i rettangoli con i lati paralleli agli assi cartesiani.

Il programma deve contenere le seguenti funzioni:

`struct punto inserisci_punto()`
: Richiede all'utente ascissa e ordinata (entrambe di tipo `int`) e restituisce la corrispondente struttura. Se una delle coordinate inserite non è compresa tra 0 e 9, essa viene richiesta di nuovo.

`struct rettangolo crea_rettangolo(struct punto a, struct punto b)`
: Riceve due punti, `a` e `b`, del piano cartesiano e restituisce il corrispondete rettangolo con i lati paralleli agli assi che ha il segmento `ab` come diagonale.

`void stampa_perimetro (struct rettangolo)`
: Stampa la regione $\mathcal{R}$ in cui è opportunamente raffigurato il perimetro di `rettangolo`.

`int main(void)`
: Richiama le altre funzioni, in modo che il programma produca i seguenti output:

```
$ ./a.out
Punto a.
Inserire l'ascissa  (valore compreso tra 0 e 9): 3
Inserire l'ordinata (valore compreso tra 0 e 9): 6
Punto b.
Inserire l'ascissa  (valore compreso tra 0 e 9): 5
Inserire l'ordinata (valore compreso tra 0 e 9): 2
9
8
7
6       * * *
5       *   *
4       *   *
3       *   *
2       * * *
1
0
  0 1 2 3 4 5 6 7 8 9
```

```
$ ./a.out
Punto a.
Inserire l'ascissa  (valore compreso tra 0 e 9): 7
Inserire l'ordinata (valore compreso tra 0 e 9): 8
Punto b.
Inserire l'ascissa  (valore compreso tra 0 e 9): 2
Inserire l'ordinata (valore compreso tra 0 e 9): 3
9
8     * * * * * *
7     *         *
6     *         *
5     *         *
4     *         *
3     * * * * * *
2
1
0
  0 1 2 3 4 5 6 7 8 9
```