Prova scritta del 9/2/2010

Strutturare adeguatamente i programmi ed evidenziarne la strutturazione mediante indentazione. Inserire anche adeguati commenti

1) Realizzare una funzione di nome `leggi_file` che, presi come suoi parametri una stringa `nome`, un intero `n` (≤ *99*), ed un array di interi `dati`, esegue le seguenti operazioni: *(a)* apre in lettura il file il cui nome è ottenuto dalla concatenazione della stringa `nome`, del numero `n` (convertito in stringa) e della stringa `".txt"` (lunghezza max. del nome risultante: *100* car.); ad es., se `nome` = `"prova"` ed `n` = `12`, il nome del file sarà: `"prova12.txt"`; *(b)* se l'apertura fallisce la funzione termina restituendo `false`; altrimenti, legge i dati presenti nel file aperto e li memorizza nell'array `dati`; la lettura termina quando si raggiunge l'`end_of_file` o i dati letti sono più di *31*; in entrambi i casi la funzione termina restituendo `true`. N.B. Usare soltanto stringhe del C. Si supponga di avere a disposizione la funzione di libreria `itoa(n,s,b)` che permette di convertire un intero `n` nella corrispondente stringa di caratteri `s`, in base `b`; ad es., se `n` = `12` e `b` = `10`, allora `s` = `"12"`.

2) Scrivere un programma principale che: *(a)* dichiara una matrice `M` di interi di dimensioni *12* ***x*** *31* e la inizializza a *0*; *(b)* legge, utilizzando (obbligatoriamente) la funzione `leggi_file`, *12* file di interi, di nome `mese1.txt`, `mese2.txt`, ..., `mese12.txt`, e li memorizza ciascuno in una riga distinta della matrice `M` (SUGG. Si richiami la funzione `leggi_file` passando come array `dati` la *i*-esima riga della matrice `M`, `M[i]`, ...); *(c)* successivamente, per ciascuna riga di `M`, calcola e stampa la somma di tutti gli elementi della riga. N.B. Nel caso l'apertura di uno dei *12* file fallisca, il programma termina immediatamente con opportuno messaggio di errore.

3) Sia `interval` il tipo di una struttura dati `struct` costituita da due campi interi, `inf` e `sup`, che rappresentano rispettivamente il limite inferiore e quello superiore di un intervallo chiuso `[inf,sup]`.
(a) Realizzare una funzione booleana di nome `disgiunti` che, presi come suoi parametri due intervalli `i1` e `i2` determina se `i1` e `i2` sono disgiunti (= hanno intersezione vuota) oppure no. Ad es., `[9,14]` e `[2,7]` sono disgiunti, mentre `[4,9]` e `[2,7]` non sono disgiunti.
(b) Realizzare una funzione di nome `fondi` che, presi come suoi parametri due intervalli non disgiunti `i1` e `i2` determina e restituisce come suo risultato il nuovo intervallo costituito dall'unione di `i1` e `i2`. Ad es., se `i1` = `[2,7]` e `i2` = `[4,9]`, il nuovo intervallo sarà `[2,9]`.
(c) Realizzare un programma principale che legge da std input due intervalli `a` e `b` e, se `a` e `b` non sono disgiunti, calcola e stampa il nuovo intervallo `c` ottenuto dall'unione di `a` e `b` (usare obbligatoriamente le funzioni `disgiunti` e `fondi`); altrimenti stampa su std output un opportuno messaggio. N.B. Controllare che l'intervallo letto sia corretto, ovvero che sia `inf` ≤ `sup`, e nel caso non lo sia, ripetere la lettura.