Esercizio 1 del 18/3

Consegnare corretto entro il 22/3 compreso

Questo esercizio richiede di leggere un numero n_el di interi in un array int X[400] e poi di essere capaci di "vedere" questo array come se fosse un array a 3 dimensioni per esempio come un array int Y[4][5][8] oppure come un array int Z[3][3][10] o ancora come int W[2][5][5].

Esempio 1: Supponiamo che n_el= 66 e che questi siano i 66 valori interi da leggere in X:

1 2 1 0 0 0 2 2 2 2 2 1 1 2 1 0 3 1 0 0 1 1 1 1 1 2 1 0 1 1 0 1

1 2 1 0 1 1 0 1 1 2 1 0 0 0 2 2 0 2 2 2 0 1 2 2 0 1 2 2 2 1 1 2 1 0

"Vedere" X come Y significa vederlo nel modo seguente:

strato 0

r0: 1 2 1 0 0 0 2 2

r1: 2 2 2 1 1 2 1 0

r2: 3 1 0 0 1 1 1 1

r3: 1 2 1 0 1 1 0 1

r4: 1 2 1 0 1 1 0 1

strato 1

r0: 1 2 1 0 0 0 2 2

r1: 0 2 2 2 0 1 2 2

r2: 0 1 2 2 2 1 1 2

r3: 1 0

Quindi Y ha solo 1 strato completamente pieno ed uno con tre righe complete e una riga finale con solo 2 elementi.

Invece "vedere" X come Z significa vederlo nel modo seguente:

strato 0

r0: 1 2 1 0 0 0 2 2 2 2

r1: 2 1 1 2 1 0 3 1 0 0

r2: 1 1 1 1 1 2 1 0 1 1

strato 1

r0: 0 1 1 2 1 0 1 1 0 1

r1: 1 2 1 0 0 0 2 2 0 2

r2: 2 2 0 1 2 2 0 1 2 2

strato 2

r0: 2 1 1 2 1 0

Cosa significhi "vedere" X come W lo lasciamo come esercizio.

Si osservi che le diverse "visioni" di X risultano in array che possono anche non essere interamente definiti. Questo succede nell'Esempio 1 sia quando vediamo X come Y che quando lo vediamo come Z. E' anche possibile "vedere" X come un array tale che il numero dei suoi elementi sia inferiore a n_el. Per esempio int K[2][2][10]. In questo caso si dovrebbero considerare solo i primi 40 elementi dei 66 definiti in X e K risulterebbe completamente pieno.

Esercizio : scrivere un programma che apra i file "input" e "output", dichiari un array int X[400] e legga da "input" il valore n_el (0<n_el<=400) e poi legga n_el valori inserendoli in X. Successivamente deve leggere da "input" tre interi positivi che sono i limiti delle 3 dimensioni dell'array a 3 dimensioni nel quale si deve "vedere" X. Se i 3 valori letti sono 4, 5 e 8, allora dobbiamo "vedere" X come Y dell'Esempio 1, mentre se leggiamo 3, 3 e 10, lo dobbiamo "vedere" come Z e così via per ogni tripla.

A questo punto il programma deve stampare su "output" gli strati della "visione" di X che viene richiesta. Questo deve venire fatto esattamente come indicato nell'Esempio 1 per Y e Z. Comprese le stringhe "strato 0", "strato 1", eccetera, all'inizio dello strato 0, 1 eccetera e anche con le stringhe "r0:", "r1:", eccetera, all'inizio della riga 0, 1 , eccetera di ciascuno strato.

Il programma deve usare due funzioni: una che si occupa di stampare uno strato (con tanto di scritta iniziale "strato n", dove n sarà 0,1 eccetera) e che a sua volta ne usa un'altra che si occupa di stampare ogni riga dello strato (con la scritta "r n:", dove n sarà 0,1,eccetera).

Per fare questo è necessario passare OUT alle funzioni e questo lo si fa prevedendo il seguente parametro formale nelle funzioni: ofstream & out. Come si vede lo stream OUT va passato per riferimento per evitare di farne una copia ad ogni invocazione. Dentro la funzione, il comando out<<x; scrive su out (che è alias di OUT) il valore di x. Si osservi che il nome out è completamente arbitrario.

Correttezza: scrivere PRE e POST di entrambe le funzioni descritte in precedenza. Dimostrare che le funzioni fanno quello che avete scritto nella loro POST.