Progetti di Laboratorio Algoritmi e Strutture Dati

Università di Bologna, corso di laurea in Ingegneria e Scienze Informatiche

Anno Accademico 2020/2021

Versione 1.0, 1 mag 2021: prima versione.

Istruzioni

Il progetto consiste in un esercizio di programmazione da realizzare in linguaggio C. La valutazione positiva del progetto è necessaria per sostenere la prova scritta. Un progetto valutato positivamente resta valido per tutti gli appelli scritti successivi, anche di anni accademici diversi.

Il progetto da consegnare è sempre quello assegnato nell'anno accademico corrente, anche per chi ha frequentato una edizione precedente del corso. In altre parole, tutti coloro che intendono consegnare il progetto per il corrente anno accademico devono fare riferimento alle specifiche descritte in questo documento.

L'esito della valutazione dipende dalla correttezza ed efficienza dei programmi consegnati, nonché dalla qualità del codice (chiarezza, nomi di identificatori scelti in modo appropriato, codice ben strutturato e comprensibile, ecc.).

Modalità di svolgimento del progetto

Il progetto deve essere svolto **individualmente**: la similitudine tra i programmi consegnati verrà verificata anche utilizzando strumenti automatici e, se confermata, comporterà l'immediato annullamento della prova per **tutti** gli studenti coinvolti. È vietato discutere le soluzioni dei progetti con gli altri studenti.

È consentito l'uso del codice messo a disposizione dal docente durante il corso. **Non è consentito fare uso di altro codice, anche se liberamente disponibile in rete**. Si tenga presente che, sebbene il docente abbia messo la massima cura nello sviluppo del software distribuito a lezione, non ne garantisce la correttezza. Ognuno sarà quindi **interamente responsabile del codice consegnato** e si assumerà la responsabilità di ogni errore, anche se l'errore era presente nel codice usato durante il corso.

I programmi devono essere conformi allo standard ANSI C (preferibile) oppure C99. Il compilatore non deve segnalare *warning* relativi a problemi facilmente risolvibili, come variabili/funzioni non utilizzate, variabili usate prima di essere inizializzate, funzioni non-void che non ritornano un risultato, eccetera. Saranno ammessi alcuni *warning* specifici di certi compilatori (es., Visual Studio che indica alcune funzioni perfettamente valide come "deprecate").

I programmi verranno compilati usando il compilatore GCC con la seguente riga di comando:

```
gcc -std=c99 -Wall -Wpedantic file.c -o file
```

(dove il nome del file corrisponde al nome del sorgente consegnato; maggiori dettagli nel seguito). Il flag `-std=c99` abilita supporto sia per C99 che per ANSI C (che è un sottoinsieme di C99).

I programmi devono produrre output a video rispettando il formato indicato in questo documento e negli esempi forniti. Potrebbero esistere più soluzioni corrette; in questi casi il programma può restituirne una qualsiasi, anche se diversa da quella fornita. Nel caso di esercizi che richiedano la stampa di valori in virgola mobile, i risultati possono variare leggermente in base all'ordine con cui vengono effettuate le operazioni, oppure dal tipo di dato utilizzato (`float` o `double`); tali differenze sono accettabili e non comprometteranno la valutazione positiva dell'elaborato.

Vengono forniti alcuni file di input con i corrispondenti risultati attesi; si può assumere che i dati di input siano sempre corretti. Si tenga però presente che **un programma che produce il risultato corretto con gli input forniti non è necessariamente corretto**. I programmi verranno testati anche con input diversi. Inoltre, come detto in precedenza, verranno valutati anche aspetti non funzionali del codice (efficienza della soluzione, chiarezza del codice, ecc.) che, se non soddisfatti, potrebbero portare ad una valutazione negativa.

È stato predisposto un forum di discussione sulla piattaforma Virtuale, sul quale è possibile chiedere chiarimenti sulle specifiche dell'elaborato (ossia, su quanto scritto in questo documento). Tutte le domande devono essere poste sul forum; **non risponderemo a mail dirette**. Ricordo che il progetto è parte dell'esame finale, e va trattato con la dovuta serietà: non risponderemo quindi a richieste di fare debug del vostro codice, né risponderemo a domande di programmazione in linguaggio C (esiste il corso di Programmazione apposta!).

Alcune buone pratiche di programmazione

All'inizio del corso sono state illustrate alcune buone pratiche di programmazione che devono essere scrupolosamente seguite. Per comodità richiamiamo le più importanti (altre sono state discusse a lezione; si faccia riferimento al materiale del corso).

- *Usare nomi appropriati per gli identificatori.* L'uso di nomi non significativi rende il codice difficile da comprendere, e potrebbe comportare una valutazione negativa.
- *Formattare correttamente il codice.* Il sorgente deve essere indentato in modo adeguato. Ciascuno è libero di usare il proprio stile di indentazione preferito, purché sia usato in modo consistente.
- *Commentare il codice in modo adeguato.* I commenti devono descrivere in modo sintetico i punti critici del codice, non parafrasarlo riga per riga.
- *Usare strutture dati adeguate.* Decidere quale struttura dati o algoritmo siano più adeguati per un determinato problema è l'obbiettivo di questo corso, e pertanto sarà un aspetto fondamentale della valutazione. Soluzioni corrette ma eccessivamente inefficienti (es., soluzioni che operano "a forza bruta") verranno respinte.
- *Evitare codice ridondante*.

Modalità di consegna

L'elaborato va consegnato tramite la piattaforma "Virtuale" in un UNICO file sorgente il cui nome deve essere il proprio numero di matricola (es., `0000123456.c`). Tutte le funzioni necessarie devono essere definite all'interno di tale file, escluse ovviamente quelle della libreria standard C. Tutti i programmi devono iniziare con un commento contenente nome, cognome, numero di matricola, gruppo (A oppure B) e indirizzo mail (`@studio.unibo.it`) dell'autore/autrice. Nel commento iniziale è possibile indicare eventuali informazioni utili alla valutazione.

Le date di consegna prima dei vari appelli d'esame sono indicate sulla piattaforme Virtuale. Dopo ciascuna scadenza non sono possibili nuove consegne fino alla data dell'esame, dopo la quale le consegne saranno riaperte fino alla scadenza successiva, e così via fino all'ultimo appello dell'anno accademico.

Modalità di valutazione

I progetti riceveranno una valutazione binaria (0=insufficiente, 1=sufficiente). I progetti valutati positivamente consentono di sostenere la prova scritta in tutti gli appelli d'esame successivi, anche di anni accademici diversi.

In caso di valutazione negativa, saranno possibili ulteriori tentativi fino al raggiungimento della sufficienza. La piattaforma Virtuale è stata configurata per consentire la consegna di una nuova versione del proprio programma non appena si riceve una valutazione negativa. Tuttavia, **in caso di riconsegna dopo la scadenza, non viene garantita la correzione in tempo utile per sostenere l'esame**. Suggerisco di consegnare con un congruo anticipo in modo da poter far fronte a eventuali richieste di correzione.

Checklist

Viene riportata in seguito un elenco di punti da controllare prima della consegna:

1. Il programma è stato consegnato in un unico file il cui nome coincide con il proprio numero di matricola?
2. È presente un commento iniziale che riporta cognome, nome, numero di matricola, gruppo (A/B) e indirizzo di posta (`@studio.unibo.it`) dell'autore?
3. Il programma produce l'output corretto sui casi di test forniti?

Esercizio: Le scatole

Disponiamo di $n \geq 1$ scatole a forma di parallelepipedo, etichettate come 0, 1, … n - 1; la scatola i ha larghezza $x[i]$, altezza $y[i]$ e profondità $z[i]$ (valori reali strettamente positivi). Vogliamo inserire il maggior numero di scatole una dentro l'altra, in modo simile ad una "matrioska": la scatola i può essere contenuta nella scatola j se le dimensioni della scatola i sono strettamente minori di quelle corrispondenti della scatola j, cioè se $x[i] < x[j]$ **and** $y[i] < y[j]$ **and** $z[i] < z[j]$.
Implementare un algoritmo efficiente per determinare il massimo numero di scatole che possono essere inserite l'una nell'altra rispettando i vincoli precedenti. Le scatole non possono essere ruotate.
Il programma accetta sulla riga di comando un unico parametro che rappresenta il nome del file di input, il cui contenuto ha la struttura seguente :

n
x[0] y[0] z[0]
…
x[*n*-1] *y*[*n*-1] *z*[*n*-1]

L'output deve elencare le scatole che fanno parte della soluzione ottima, una scatola per riga, a cominciare da quella più esterna; per ciascuna di esse devono essere riportate le dimensioni (come da file di input).

Esempio.

Input:	**Ouptut:**
10 10 3 4 2 2 3 3 2.5 3 4.4 5 6 7 5.1 7.4 9 8.7 5.6 8.7 6.5 9.5 2.5 6.5 7.3 5.7 8.7 9.8 7.6 5.1 6.2	4 scatole scatola 6: 8.7 6.5 9.5 scatola 4: 7.0 5.1 7.4 scatola 3: 4.4 5.0 6.0 scatola 1: 2.0 2.0 3.0

Nell'esempio precedente vengono fornite 10 scatole (che verranno numerate da 0 a 9). Il risultato indica che è possibile inserire al massimo 4 scatole una dentro l'altra: la scatola 6 contiene la 4, che contiene la 3, che a sua volta contiene la 1.
Potrebbero esistere più soluzioni ottime, cioè più soluzioni diverse che coinvolgono lo stesso numero di scatole; il programma verrà considerato corretto se stampa una qualunque di tali soluzioni.