Progetto di Laboratorio Algoritmi e Strutture Dati

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

Anno Accademico 2024/2025

Versione 1.0, 13 maggio 2025

Istruzioni

Questo documento descrive le specifiche e le modalità di svolgimento e valutazione.

Modalità di svolgimento del progetto

I programmi devono essere conformi ANSI C (detto anche C89 o C90), e verranno compilati usando GCC con la seguente riga di comando:

```
gcc -std=c90 -Wall -Wpedantic file.c -o file -lm
```

(dove il nome del file verrà sostituito dal nome del sorgente consegnato; maggiori dettagli nel seguito). Il flag `-lm` serve per linkare la libreria matematica, nel caso in cui qualcuno desideri usare qualche funzione dichiarata in `math.h` (che comunque non dovrebbe essere necessaria). Il compilatore non deve segnalare né errori né *warning*, soprattutto se relativi a problemi facilmente risolvibili come variabili/funzioni non utilizzate, variabili usate prima di essere inizializzate, funzioni non-void in cui non si restituisce un risultato, eccetera. Saranno ammessi alcuni *warning* specifici di Visual Studio (es., quelli relativi alla sostituzione di `fopen` con `fopen_s` e simili; tali warning devono essere ignorati, dato che `fopen_s` non fa parte di ANSI C).

La funzione `main()` deve restituire 0 (oppure `EXIT_SUCCESS`) al termine dell'esecuzione corretta; è lasciata libertà di restituire un codice di errore nel caso in cui il programma termini in modo anomalo. Tuttavia, non si dovrebbero mai verificare terminazioni anomale perché gli input sono garantiti essere sempre corretti.

I programmi devono produrre output a video rispettando scrupolosamente il formato indicato in questo documento e negli esempi forniti. I programmi verranno inizialmente controllati in modo semi-automatico, e respinti in caso di output non conforme alle specifiche. Vengono forniti alcuni file di input con i corrispondenti risultati attesi; in certi casi possono esistere più soluzioni ottime, e i programmi verranno considerati corretti se ne calcolano una qualsiasi (maggiori informazioni in seguito). Si tenga 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.

I programmi non devono interagire in alcun modo con l'utente: non devono eseguire comandi di sistema (es., `system("pause")`), né richiedere all'utente di premere invio, inserire informazioni da tastiera, o altro.

I programmi non devono fare uso di file temporanei: tutte le informazioni ausiliarie eventualmente necessarie devono essere tenute in memoria in apposite strutture dati.

I programmi non devono presentare accessi “out-of-bound” né altri tipi di accessi non validi alla memoria; devono liberare in modo esplicito con `free()` tutta la memoria allocata con `malloc()` prima di terminare; in caso di terminazione a causa di condizioni anomale (che comunque non si dovrebbero mai verificare con gli input forniti), non è necessario liberare la memoria. Questi aspetti sono importanti perché possono produrre problemi non immediatamente evidenti (es, gli accessi “out-of-bound” potrebbero causare un crash con certe combinazioni di compilatore e sistema operativo, e non con altre), per cui è necessaria la massima attenzione. Per verificare l'uso corretto della memoria verrà usato il programma *valgrind* in ambiente Linux. In particolare, i programmi verranno esaminati con il seguente comando:

```
valgrind ./nome_eseguibile nome_file_di_input
```

che non deve segnalare errori. Ricordarsi di chiudere il file di input con `fclose()`, dato che in caso contrario *valgrind* segnalerà memoria non liberata (si tratta di memoria che viene riservata internamente da `fopen()` e liberata da `fclose()`).

I programmi devono rispettare le indicazioni contenute nella dispensa *Programmazione: Breve guida di stile* disponibile sulla pagina del laboratorio, il cui contenuto fa parte integrante di questa specifica. Verranno quindi valutati anche aspetti non funzionali (efficienza, chiarezza del codice, ecc.) che, se non soddisfatti, potrebbero portare ad una valutazione negativa e la necessità di modificare il sorgente e riconsegnare.

Sulla piattaforma Virtuale è stato predisposto un forum di discussione, nel quale è possibile chiedere chiarimenti sulle specifiche dell'elaborato (ossia, su questo documento); tutte le domande devono essere poste esclusivamente sul forum. Poiché il progetto è parte dell'esame finale, va trattato con la dovuta serietà: di conseguenza, non faremo debug del codice né risponderemo a quesiti di programmazione in C. Si assume che chi ha seguito questo corso sappia già programmare in C (che come detto all'inizio del corso, è un prerequisito).

Il progetto deve essere svolto individualmente; non è consentito discutere il progetto o le soluzioni con altri studenti o con terzi. Il rispetto di tali vincoli verrà verificato con strumenti automatici e, in caso di violazione, comporterà l'annullamento delle consegne per tutti gli studenti coinvolti, e successiva segnalazione all'ufficio di Ateneo per le sanzioni disciplinari che potrà comminare ulteriori sanzioni. Verrà inoltre assegnato un nuovo progetto basato su nuove specifiche, che potrà essere consegnato solo dopo l'appello d'esame successivo.

Come unica eccezione al punto precedente, è consentito l'uso del codice messo a disposizione dal docente durante il laboratorio, incluse le soluzioni degli esercizi proposti; è possibile apportare qualunque tipo di modifica si ritenga necessaria. Si tenga presente che, sebbene sia stata messa la massima cura nello sviluppo dei programmi forniti in laboratorio, non se ne garantisce la correttezza. Ognuno/a sarà quindi interamente responsabile del codice consegnato e si assumerà la responsabilità di ogni errore, anche se presente nel codice fornito dal docente. **Non è consentito fare uso di altro codice, anche se liberamente disponibile in rete, né l'uso di IA generativa** (verranno usati appositi software per identificare codice prodotto da IA): chi usa *code assistant* si espone al rischio di consegnare codice (spesso sbagliato) molto simile o del tutto identico a quello di altri, che verrà valutato come copiato.

Non è consentito condividere la soluzione di questo esercizio con terzi, ad esempio rendendola disponibile su un repository pubblici o comunicandola con altri mezzi.

I docenti si riservano la facoltà di chiedere modifiche al codice consegnato, anche se funzionante, e/o di richiedere chiarimenti tramite una discussione orale.

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 devono essere definite all'interno di tale file, escluse quelle della libreria standard C. All'inizio del sorgente deve essere presente un commento in cui si indica nome, cognome, numero di matricola, classe (A oppure B) e indirizzo mail (`@studio.unibo.it`) dell'autore/autrice.

Le date di consegna 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.

Valutazione

I programmi verranno valutati dopo la chiusura delle consegne, e 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: **i progetti valutati positivamente non scadono mai**, e pertanto non bisognerà riconsegnare il progetto anche se si sostiene l'esame in anni accademici successivi.

Attenzione: sarà possibile presentarsi al **primo** appello scritto di giugno anche senza aver consegnato il progetto; in tal caso, è obbligatorio consegnare il progetto (e ricevere una valutazione positiva) entro la scadenza indicata per il secondo appello scritto. A partire dal secondo appello scritto, sarà possibile presentarsi solo dopo aver consegnato il progetto entro la scadenza indicata su Virtuale e aver ricevuto valutazione positiva.

Ai progetti valutati negativamente che, a giudizio dei docenti, presentino errori facilmente risolvibili, verrà concesso un ulteriore tentativo. In caso di nuova consegna insufficiente, o nel caso in cui la consegna originale presenti errori gravi, sarà possibile una nuova consegna solo dopo la prova scritta.

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 è 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 è conforme allo standard ANSI C (detto anche C89 o C90).
4. Il programma compila senza errori né *warning* usando la riga di comando di GCC indicata in questa specifica.
5. Il programma è conforme alle specifiche; l'output sui casi di test forniti è corretto e rispetta scrupolosamente il formato indicato in questo documento e nei file di output forniti.
6. Il programma libera tutta la memoria allocata con `malloc()` in caso di terminazione corretta. È stata controllata l'assenza di *memory leak* e accessi *out-of-bound*.

Shooting Stars

All'inizio del corso abbiamo visto il gioco *Shooting Stars*; si rimanda alla pagina dell'esercizio per i dettagli. Scopo di questo progetto è implementare un algoritmo efficiente per determinare la sequenza minima di mosse a partire da una configurazione iniziale data in input. Si noti che in questo progetto la configurazione iniziale è un parametro del problema, mentre nell'esercizio originale si assumeva una configurazione iniziale fissa.

Vincoli

- Nel caso si usino liste, code o stack, è obbligatorio realizzarle mediante nodi e puntatori, non array. Si può utilizzare il tipo di dato *List* visto in laboratorio, eventualmente apportando le necessarie modifiche.

Input

Il programma accetta come unico parametro sulla riga di comando il nome del file di input contenente la configurazione iniziale. Il file contiene tre righe, ciascuna con tre caratteri che possono essere il punto oppure l'asterisco; l'asterisco corrisponde ad una stella, mentre il punto rappresenta un buco nero.

Output

Al termine dell'esecuzione, il programma deve stampare a video la sequenza minima di mosse che portano alla configurazione vincente, che ricordiamo essere la seguente:

```
***
*.*
***
```

La sequenza di mosse deve essere stampata come la sequenza dei numeri corrispondenti alle stelle da fare "esplodere" (un numero per ogni riga), rispettando le regole del gioco. Le caselle sono numerate come già visto:

```
012
345
678
```

Nel caso in cui la configurazione vincente non possa essere ottenuta da quella iniziale, il programma deve stampare -1

Esempi

Contenuto del file di input	*Output stampato a video*

`*.*`
`**.`
`..*`
	`0`
	`2`
	`4`
	`1`
	`3`
	`6`
	`3`
	`4`

Contenuto del file di input	*Output stampato a video*

`*..`
`...`
`...`
	`0`
	`1`
	`0`
	`2`
	`5`
	`2`
	`1`
	`8`
	`7`
	`4`

`...` `...` `...`	`-1`

Note

Possono esistere più sequenze di lunghezza minima che conducono alla configurazione vincente; in tali casi è sufficiente stamparne una qualunque, anche se diversa dalla soluzione fornita. Tuttavia, il numero di mosse deve essere uguale a quello della soluzione fornita.