Progetto di esempio

Il seguente progetto è riferito all'anno accademico precedente; non è quindi il progetto d'esame del corrente anno accademico, ma viene reso disponibile solo come esempio.

Istruzioni

Il progetto consiste in un esercizio di programmazione da realizzare in ANSI C. Questo documento descrive le specifiche del progetto e le modalità di svolgimento e valutazione.

Modalità di svolgimento del progetto

Il progetto deve essere svolto **individualmente**; non è consentito discutere le soluzioni con altri studenti o con terzi. La similarità tra consegne diverse verrà valutata con strumenti automatici e, se confermata, comporterà l'annullamento delle consegne per **tutti** gli studenti coinvolti e la necessità di consegnare un nuovo progetto su specifiche diverse.

È consentito l'uso del codice messo a disposizione dal docente durante il laboratorio, incluse le soluzioni degli esercizi. **Non è consentito fare uso di altro codice, anche se liberamente disponibile in rete**. Si tenga presente che, sebbene sia stata messa la massima cura nello sviluppo dei programmi distribuiti a lezione, non se ne garantisce la correttezza. Ognuno sarà quindi **interamente responsabile del codice consegnato** e si assumerà la responsabilità di ogni errore, anche se presente nel codice fornito dal docente.

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

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

(dove il nome del file verrà sostituito dal nome del sorgente consegnato; maggiori dettagli nel seguito). I programmi devono essere conformi allo standard ANSI C (detto anche C89 oppure C90). Il compilatore non deve segnalare *warning*, soprattutto se 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 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).

I programmi consegnati devono produrre output a video **rispettando scrupolosamente il formato indicato in questo documento e negli esempi forniti**. I programmi verranno inizialmente verificati in modo semi-automatico, e rigettati in caso di output non conforme alle specifiche.

I programmi verranno testati prevalentemente in ambiente Ubuntu Linux, ma devono funzionare correttamente anche su Windows e MacOSX. I programmi **non devono interagire in alcun modo con l'utente**: non devono eseguire comandi di sistema (es., `system("pause")`), né devono richiedere all'utente di premere invio, inserire informazioni da tastiera, o altro.

In fase di valutazione verrà prestata particolare attenzione alla corretta gestione della memoria. In particolare, verrà verificato che i programmi non presentino accessi "out-of-bound" e che tutta la memoria allocata con malloc() venga esplicitamente liberata con free() prima della terminazione del programma. Questi aspetti sono molto insidiosi perché possono risultare 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 in fase di sviluppo e test.

Vengono forniti alcuni file di input con i corrispondenti risultati attesi. Si può assumere che tutti gli input con cui verranno testati i programmi siano sempre corretti; non è quindi necessario includere controlli (sebbene sia comunque una buona pratica di programmazione). 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. Verranno inoltre valutati anche aspetti non funzionali del codice (efficienza, chiarezza del codice, ecc.) che, se non soddisfatti, potrebbero portare ad una valutazione negativa.

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 sul forum. Poiché il progetto è parte dell'esame finale, esso va trattato con la dovuta serietà: di conseguenza, **non faremo debug del codice né risponderemo a quesiti di programmazione in C**.

Alcune buone pratiche di programmazione

All'inizio del corso è stata fornita una dispensa che illustra alcune buone pratiche di programmazione. Il rispetto delle regole indicate nella dispensa è obbligatorio (ma non sufficiente) per ottenere una valutazione positiva. Per comodità richiamiamo le regole più importanti (occorre comunque rispettare tutto quanto scritto nella dispensa).

- *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. Le righe non devono essere più lunghe di 80 caratteri (se necessario, andare a capo).
- *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. Verranno respinte soluzioni corrette ma eccessivamente inefficienti.
- *Non scrivere tutto il programma nel main()*: decomporre il problema in passi distinti da delegare ad apposite funzioni.

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 che indichi nome, cognome, numero di matricola, gruppo (A oppure B) e indirizzo mail (`@studio.unibo.it`) dell'autore/autrice.

Le date di consegna sono indicate sulla piattaforme Virtuale, e sono indicativamente 10 giorni prima di ogni appello scritto. 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 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 (i progetti valutati positivamente "non scadono").

In caso di valutazione negativa sarà possibile modificare il programma e riconsegnarlo. Si presti però attenzione al fatto che **in caso di riconsegna dopo la scadenza, non viene garantita la correzione in tempo utile per sostenere l'esame**.

Checklist

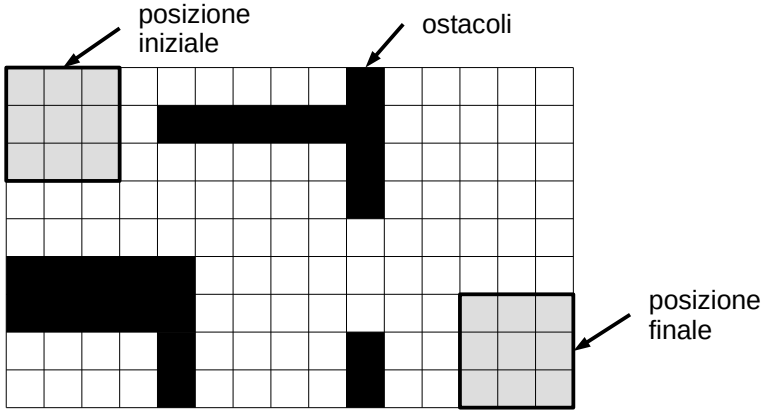
Viene riportata in seguito un elenco di punti da controllare prima della consegna:

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 è conforme allo standard ANSI C (detto anche C89 o C90)?
4. Il programma compila senza *warning*?
5. Il programma è conforme alle specifiche? L'output è corretto e rispetta scrupolosamente il formato indicato in questo documento e nei file di esempio?
6. Il programma libera correttamente la memoria prima di terminare? È stata verifica l'assenza di accessi out-of-bound o altri errori connessi all'uso della memoria, ad esempio usando il tool “valgrind” in ambiente Linux?

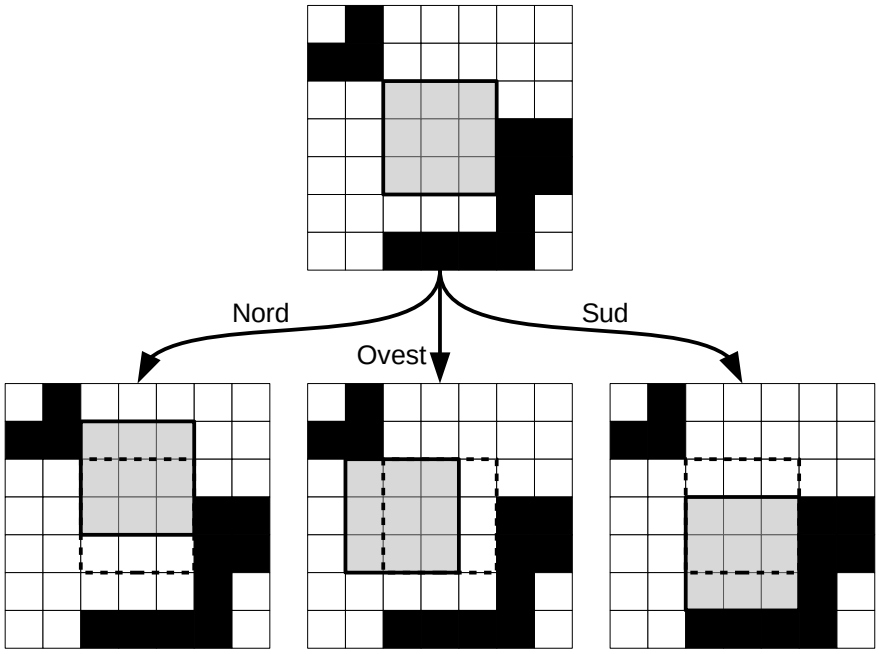
Il robot aspirapolvere

Un robot aspirapolvere deve raggiungere la postazione di ricarica evitando gli ostacoli. Scopo di questo esercizio è progettare e realizzare un algoritmo efficiente per decidere se la postazione di ricarica è raggiungibile, e in caso affermativo determinare il minimo numero di spostamenti necessari per arrivarci e il percorso da seguire.

La mappa della stanza è descritta da una matrice di caratteri composta da *n* righe e *m* colonne. Il robot ha un ingombro di 3 × 3 celle, ed è inizialmente posizionato nell'angolo in alto a sinistra; la postazione di ricarica si trova nell'angolo in basso a destra.



Ad ogni passo il robot è in grado di spostarsi di una cella nelle quattro direzioni (Nord, Sud, Est, Ovest), purché la propria area di ingombro non contenga ostacoli. L'area di ingombro deve sempre rimanere interamente contenuta all'interno della matrice; si può quindi assumere che la stanza sia circondata da un muro. Nell'esempio seguente, il robot può effettuare uno spostamento di un passo lungo le tre direzioni nord, ovest, sud; non può spostarsi a est a causa della presenza di ostacoli.



Il programma deve leggere l'input da un file il cui nome completo viene passato come unico parametro sulla riga di comando. Il file ha il seguente formato:

- La prima riga contiene i valori *n*, *m* separati da spazi o tab;
- Seguono *n* righe, ciascuna composta da *m* caratteri (più un ritorno a capo finale); ogni carattere può essere il punto (.) che denota una cella libera, oppure l'asterisco (*) che denota una cella occupata da un ostacolo.

Il programma deve stampare a video:

- Il numero minimo K di spostamenti necessari per raggiungere la postazione di ricarica partendo dalla posizione iniziale; se la postazione di ricarica non è raggiungibile, il programma deve stampare -1;
- Solo se la postazione di ricarica è raggiungibile, il programma deve stampare una seconda riga contenente K caratteri che indicano gli spostamenti effettuati. Ogni carattere può essere 'N' (spostamento verso nord), 'S' (sud), 'E' (est), 'O' (ovest). Se la postazione di ricarica non è raggiungibile, la seconda riga non viene stampata.

Assunzioni:

- $10 \leq n, m \leq 1000$
- Gli elementi della sottomatrice 3×3 posizionata nell'angolo in alto a sinistra contengono tutti '.', quindi non ci sono ostacoli nella posizione iniziale del robot;
- Gli elementi della sottomatrice 3×3 posizionata nell'angolo in basso a destra contengono tutti '.', quindi non ci sono ostacoli nella posizione di destinazione del robot;
- Possono esistere più soluzioni ottime, corrispondenti allo stesso valore di K ma a diverse sequenze di spostamenti. È sufficiente stamparne una qualunque, anche se diverso da quella riportata negli output di esempio; il valore di K deve invece essere lo stesso.

Esempi:

Input:	*Output:*
10 10 ...******* ...*...... ....*... ...****... ......*... ......*... *******...	14 SSEEEEEEESSSSS
Input: 15 30 ....**********.....**... ....**********.......... ....**.......... ..................**.....**... ..................**.....**... ....**............**.....**... **************....**.....**... **************....**.....**... ....**............**.....**... ....**............**.....**... ....**....**********.....**... ....**....**********.....**... ******..................**... ....**... ....**...	*Output:* -1