Corso di Laboratorio di Programmazione
Progetto finale – Battaglia Navale

Vi viene richiesto di sviluppare un programma in C++ che implementi una versione avanzata del gioco Battaglia Navale. Le regole sono state modificate quindi prestate attenzione alla descrizione del gioco.

Regole del gioco

Ogni giocatore possiede due griglie di gioco composte da 12x12 caselle. Una griglia di difesa che serve per posizionare le proprie unità navali, e una griglia di attacco che serve per tenere traccia dei colpi alle unità avversarie andati a segno (X) o meno (O). Inoltre ogni giocatore possiede 8 unità di tre tipologie diverse. I tre tipi di unità sono: corazzata (C), nave di supporto (S), sottomarino di esplorazione (E). Di seguito un esempio di griglie di gioco:

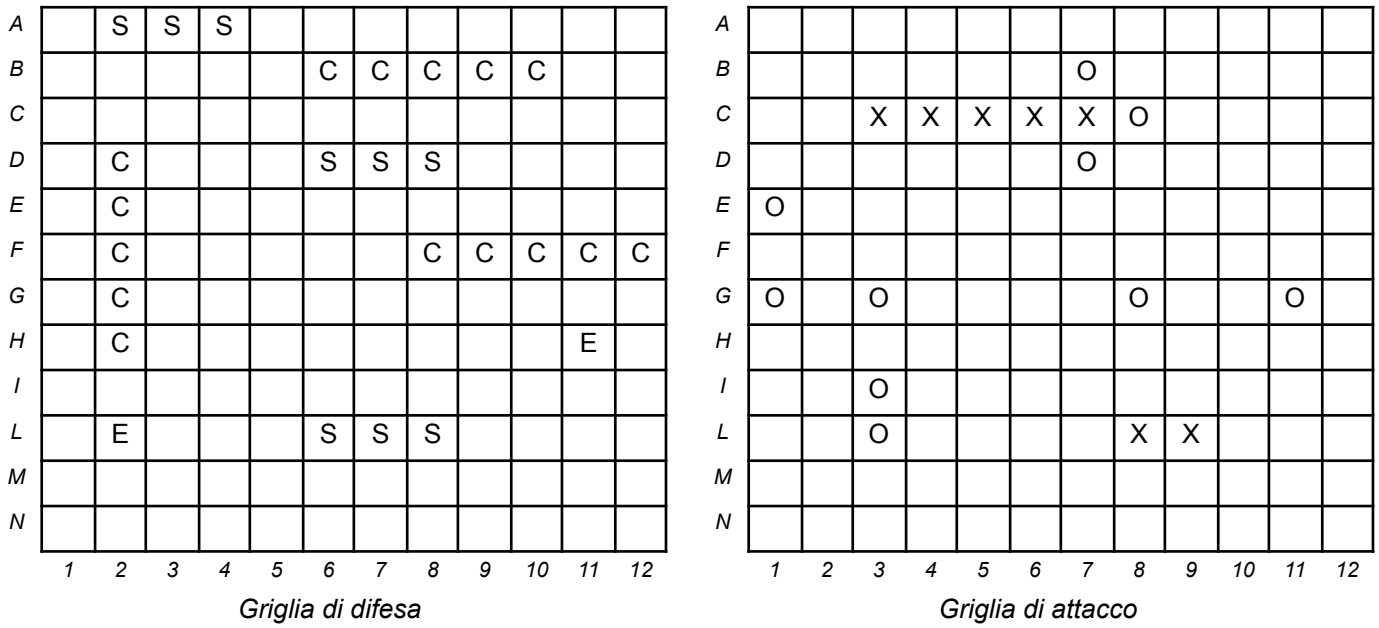

Figura 1. Esempio di griglia di difesa (a destra) e di attacco (a sinistra)

Unità navali

Tutte le unità navali possono compiere un'***azione*** che però si differenzia a seconda del tipo di unità. Inoltre ogni unità ha una dimensione specifica che equivale anche alla sua corazza iniziale. Quando tutte le caselle che un'unità occupa vengono colpite, l'unità è da considerarsi affondata. Corazza e dimensione sono inizialmente equivalenti. Man mano che l'unità viene colpita la corazza diminuisce mentre la dimensione rimane la stessa. Di seguito vengono riportate le caratteristiche di ogni unità navale:

Unità navale	Quantità	Dimensione	Corazza	Azione
Corazzata	3	5	5	Fuoco
Nave di supporto	3	3	3	Muovi e ripara
Sottomarino di esplorazione	2	1	1	Muovi e cerca

Le ***azioni*** che le diverse unità navali possono eseguire sono riportate di seguito:

- Fuoco: la **corazzata** fa fuoco su determinate coordinate. Il colpo va a segno se la casella colpita è occupata da un'unità avversaria. Se la corazza dell'unità bersaglio raggiunge 0 (tutte le caselle in cui l'unità è presente sono state colpite), l'unità bersaglio viene affondata

- Muovi e ripara: la **nave di supporto** si sposta sulle coordinate ricevute. Ovviamente la casella non deve essere già occupata. Una volta che si è mossa, dà supporto a tutte le unità del giocatore presenti in un'area 3x3 considerando come centro la stessa nave di supporto. Le unità in quest'area verranno completamente riparate (cioè la corazza ritorna al valore iniziale). Basta che una sola casella sia all'interno dell'area perché tutta l'unità riceva supporto. La nave che dà supporto non può beneficiare delle sue stesse riparazioni (quindi per essere riparata deve beneficiare dell'aiuto di un'altra nave di supporto).

- Muovi e cerca: il **sottomarino** si sposta sulle coordinate ricevute. Ovviamente la casella non deve essere già occupata. Una volta che si è mosso, lancia un impulso sonar che svela se ci sono unità avversarie in un'area 5x5 considerando come centro il sottomarino stesso. Le caselle in cui ci sono unità avversarie (o parti di unità avversarie) vengono riportate sulla griglia di attacco con il carattere Y.

Nota sul movimento:
Il movimento è relativo alla casella centrale dell'unità. Ogni movimento deve essere valido, cioè l'unità dopo il movimento deve essere interamente all'interno della griglia e non può sovrapporsi con altre unità del giocatore. Non esiste rotazione, le unità traslano in modo rigido sulla griglia.

Giocatori e turni

Esistono sempre e solo due giocatori. Uno dei due giocatori può essere umano. All'inizio del gioco, tutti i giocatori posizionano le unità sulla propria griglia di difesa. Il programma decide in modo casuale il giocatore che inizia il gioco. Nel proprio turno, il giocatore deve far compiere un'***azione*** a solo una delle proprie unità. Una volta che l'***azione*** è compiuta, il turno passa all'altro giocatore.

Nota sull'intelligenza del giocatore computer:
Non vi chiediamo di implementare degli algoritmi intelligenti per il computer. Nel turno, il giocatore computer seleziona in modo casuale una delle proprie unità e ne esegue l'***azione***.

Comandi di gioco

Ogni giocatore fa eseguire l'***azione*** a una delle proprie unità, utilizzando un comando con la seguente sintassi:

```
XYOrigin XYTarget
```

`XYOrigin` corrispondono alle coordinate dell'unità che deve compiere l'azione (le coordinate che identificano un'unità sono quelle della casella centrale). `XYTarget` corrispondono alle coordinate che sono target dell'azione dell'unità. In caso di corazzata corrispondono alla casella target di fuoco; nel caso di nave di supporto e sottomarini di esplorazione corrispondo alla casella di arrivo dello spostamento.
Prendendo come esempio, la Figura 1, alcuni esempi di comandi:

`B8 G10` -> Corazzata in B8 fa fuoco su casella G10
`L7 I2` -> Nave di supporto in L7 si sposta in I2 e fornisce assistenza (ripara) l'unità in F2 e L2

Con un comando speciale (`AA AA`) è possibile cancellare gli avvistamenti sonar (Carattere Y) nella griglia di attacco (in quanto le unità si sono mosse, e gli avvistamenti possono non essere aggiornati). Questo comando non usa la mossa del turno.

Posizionamento iniziale delle unità navali

Il posizionamento iniziale delle unità navali viene effettuato a inizio partita. In questa fase di gioco, il programma chiede al giocatore di fornire le coordinate della prua e della poppa di ognuna delle proprie unità. Le unità possono essere disposte solo in senso orizzontale o verticale e non possono sovrapporsi (nemmeno nel caso di sottomarini, per semplicità). Ad esempio (facendo riferimento alla Figura 1):

```
>> Quali sono le coordinate per la corazzata 1:
>> B6 B10
>> Quali sono le coordinate per la corazzata 2:
>> D2 H2
…
```

Visualizzazione

Nel caso di partita con un giocatore, è possibile richiedere al programma la visualizzazione delle griglie di difesa e di attacco usando un comando speciale:

`XX XX`

La visualizzazione avviene stampando un carattere per ciascuna unità navale nella griglia di difesa e per ciascun colpo effettuato e per le rilevazioni sonar nella griglia di attacco, come riportato in Figura 1. Ogni griglia deve avere le coordinate come riportato in Figura 1. I caratteri sulle griglie hanno la seguente codifica:

Griglia di difesa	
Corazzata	C
Nave di supporto	S
Sottomarino di esplorazione	E

Griglia di attacco	
Unità colpita	X
Acqua	O
Rilevazione sonar	Y

Se non è presente nessuna unità o nessuno colpo effettuato o nessuna rilevazione, è stampato uno spazio. Le caselle corrispondenti a parti di unità colpite sono segnate con lettera minuscola.

Partite

Le partite sono di due tipi:

- partite computer vs. computer;
- partite giocatore vs. computer.

Il programma deve gestirle entrambe. Nel caso di partite tra due computer, dovete fissare un numero massimo di turni, oltre il quale la partita è considerata nulla. Durante ogni partita deve essere effettuato un log su file, che elenca tutti i comandi inviati, in ordine. Il programma deve essere in grado di effettuare il replay di una partita dato un log su file, che deve essere un file testuale con estensione .txt.

Replay

Il progetto deve essere corredato di un modulo (costituito da un eseguibile a parte, vedi prossimo punto) per il replay della partita, effettuato leggendo il relativo log. Il replay è realizzato stampando le due griglie di gioco per ogni turno. La stampa avviene a video, con un'opportuna pausa (es., 1 secondo) tra una turno e il successivo, oppure su file – in questo caso la scrittura avviene senza pause.

Eseguibili

Il progetto deve generare due eseguibili:

- l'eseguibile del gioco, chiamato "battaglia_navale", che deve gestire gli argomenti da riga di comando, secondo il seguente schema:
  - argomento pc: partita giocatore vs. computer (p sta per player, c per computer);
  - argomento cc: partita computer vs. computer;
- l'eseguibile per il replay, chiamato "replay", che accetta gli argomenti da riga di comando secondo il seguente schema:
  - argomento v [nome_file_log]: stampa a video il replay del file di log indicato;
  - argomento f [nome_file_log] [nome_file_output_replay]: scrive su file il replay del file di log indicato.

Note allo sviluppo

Il progetto sviluppato **deve** essere gestito tramite CMake e compilabile sulla macchina virtuale Taliercio.2020. Non è necessario che sviluppiate tutto sulla macchina virtuale, ma dovete fare verifiche periodiche per essere sicuri di non aver utilizzato codice fuori standard.

Indicazioni per lo svolgimento

Il progetto deve essere suddiviso in più file sorgente. **Ogni file deve essere scritto da un solo studente**. È tuttavia possibile controllare che il codice dei propri compagni di gruppo funzioni correttamente. Un errore in un file che inficia il funzionamento del progetto potrebbe causare una penalizzazione della valutazione dell'intero gruppo. **Il nome dell'autore deve essere**

indicato in un commento a inizio file. Ovviamente, è necessario e positivo che si discuta all'interno di ciascun gruppo su come realizzare il codice, ma ogni studente è responsabile della gestione del codice che deve scrivere.

Saranno valutati:

- Chiarezza e correttezza del codice;
- Corretta gestione della memoria e delle strutture dati;
- Efficienza del codice e della soluzione trovata;
- Utilizzo di strumenti appropriati.

Il codice deve essere adeguatamente commentato.

Il software deve essere basato unicamente sulla libreria standard del C++.

Plagi

Il codice verrà controllato per verificare eventuali plagi tra gruppi appartenenti allo stesso canale e in canali diversi. Verrà effettuato anche un controllo rispetto a codice presente e disponibile online.

Consegna

Il compito deve essere consegnato su Moodle (consegna di gruppo come per la prova intermedia), caricando un archivio che include una sola directory contenente:

- Il codice sorgente (eventualmente organizzato in sottodirectory);
- Il file CMakeLists.txt necessario alla compilazione (uno solo e posto nella directory principale del progetto);
- Un file di log di una partita computer vs. computer e un file di log di una partita computer vs giocatore;
- Un file readme.txt in cui riportate eventuali note che volete comunicare in fase di correzione. Questo file non è la documentazione, che deve essere inserita sotto forma di commenti nel codice, ma un elenco di note aggiuntive per chi corregge, per esempio problemi riscontrati e non risolti.

L'archivio non deve contenere l'eseguibile, perché il sorgente sarà compilato in fase di correzione. Il sistema CMake deve compilare con le opzioni di ottimizzazione attivate (-O2).
Dopo la consegna su moodle, **verificate ciò che avete consegnato** con i passi seguenti:

- Scaricate il vostro progetto da moodle in una directory diversa da quella usata per sviluppare;
- Lanciate cmake;
- Compilate il codice e verificate la corretta esecuzione.

Si suggerisce di consegnare anche compiti non completi. È tuttavia necessario che il software consegnato sia compilabile ed eseguibile.

La consegna in ritardo degli elaborati sarà **fortemente penalizzata** e considerata solo in un brevissimo lasso temporale dopo la scadenza (pochi minuti). Consegne effettuate successivamente saranno ignorate.