**I.T.T.S.**

**“S.Fedi-E.Fermi”**

Informatica

2017/2018

3 IB

*Vestri Luigi*

*Caramelli David*

**Batnav**

**Bujor Marcel Iulian**

**Galligani Stefano**

**11/04/18-15/05/18**

[**Battaglia Navale**](#_xuy5tczdesyo) **2**

[Scopo](#_gtp7uvtnfbl8) 3

[Analisi funzionale](#_rorwxgyfm15r) 3

[Analisi tecnica](#_pwv84wegf4nl) 4

[Manuale per l’utente](#_kfpciri0f529) 4

[Obiettivo](#_6sl0llu1ewpn) 4

[Avvio](#_acg4md79zib6) 4

[Screenshot del programma in esecuzione](#_opm6s2q7l0fr) 4

[Flusso del programma](#_x2euj7p7p6xi) 4

[Requisiti tecnici](#_qilxqstih6vj) 5

[Test data set\Debug](#_qx5pqpejmiya) 5

[Bibliografia\Sitografia di riferimento](#_2ppc859ynwj3) 5

[Osservazioni e conclusioni personali](#_k03m074y3dhd) 5

# **Battaglia Navale**

Si vuole scrivere un programma che realizza il gioco della “battaglia navale” tra due giocatori.

Lo scopo del gioco consiste nel riuscire a individuare prima dell’avversario la dislocazione di tutte le navi nemiche.

Ogni giocatore una volta disposte su uno scacchiere di 10 x 30 le proprie navi passa alla fase di gioco che consiste nell’indicare a turno le coordinate di un punto della scacchiera dove si ritiene che si trovi la nave avversaria.

Le navi sono caratterizzate da una descrizione e da una dimensione in caselle che

ne indica la lunghezza sullo scacchiere (Es. cacciatorpediniere da 5 caselle). Le navi possono essere disposte orizzontalmente o verticalmente ma non diagonalmente.

Il gioco si presenta con il seguente menu:

1. Gestione navi
2. Inizio Gioco
3. Classifica
4. Esci

**Gestione navi**

Il programma propone uno o più elenchi di navi predefiniti. L'obiettivo è di configurare la flotta delle navi che ogni giocatore deve disporre sul proprio scacchiere.

**Inizio Gioco**

Il gioco si articola in due fasi: preparazione e battaglia. Nella prima fase ciascun giocatore imputa il proprio nickname e dispone le navi configurate all’interno dello scacchiere (Scacchiere di difesa).

Il giocatore avversario non deve vedere dove il nemico dispone le navi. Nella seconda fase ogni giocatore alternativamente indica le coordinate del campo avversario dove suppone ci sia una nave e che verranno evidenziate nelle scacchiere d’attacco. Il programma verifica automaticamente e informa il giocatore se ha centrato o meno l'obiettivo. Nel caso che una nave sia colpita in tutti i suoi elementi (caselle), il programma risponde “colpita e affondata”.

Non appena uno dei due giocatori ha affondato tutte le navi dell’avversario il programma gli assegna la vittoria e visualizza tutte e 4 gli scacchieri di gioco: 2 di attacco e 2 di difesa.

**Classifica**

Il programma tiene traccia dei primi dieci giocatori vittoriosi che hanno totalizzato i migliori punteggi mediante memorizzazione su file.

La graduatoria è stilata in base alla dimensione della flotta, che è uguale alla sommatoria delle caselle di ciascuna nave, e a parità di dimensioni, al tempo impiegato.

## Scopo

Realizzare un programma che consenta di effettuare n partite per k giocatori, giocando due per volta.

La lista dei vincitori generale è composta dai migliori dieci vincitori.

## Analisi funzionale

Vincoli aggiuntivi:

* Le griglie non avranno dimensioni di 10X30 come richiesto dal problema, perché ritenuto troppo complesso da realizzare, considerando la risoluzione variabile degli schermi dei pc su cui il gioco potrà essere avviato. In compenso le dimensioni della griglia saranno personalizzabili con valori da 10 a 20 compresi.

Nel gioco saranno presenti:

* Un menù, per leggere il regolamento, che include le informazioni sul gioco, sulle navi e sulle griglie, ed eventualmente rimuovere i suoni, con possibilità di rimuovere solo quelli di sottofondo o solo quelli speciali (mancato-colpito-affondato);
* La possibilità di scegliere il nome per ogni giocatore, a gruppi di due, i nomi dovranno essere diversi sia l’uno dall'altro che dai vincitori della classifica;
* Possibilità di scegliere la grandezza delle griglie e scelta tra alcuni elenchi di navi predefiniti o la creazione di un elenco personalizzato;
* Una griglia per la difesa e una per il radar, dove verranno visualizzati i colpi fatti con specificato da colori se mancato, blu, o colpito, rosso;
* Un elenco degli ultimi colpi di ciascun giocatore;
* Disposizione delle proprie navi, con possibilità di modificane le posizioni. Sarnno chieste le coordinate della nave, l’orientamento, escluso per le navi da 1, e un controllo in modo che le navi non vengano sovrapposte;
* Fase d’attacco, dove il giocatore di turno indicherà le coordinate del suo colpo, con controllo che non sia un colpo già fatto;
* Messaggio di fine turno, con la richiesta di premere un tasto per continuare, dove non verranno visualizzate le griglie, in modo che i giocatori si possano intercambiare nel gioco senza rischiare di intravedere la griglia avversaria;
* Visualizzazione finale con tutte e quattro le griglie, due attacco e due radar, con possibilità di fare un altra partita o concludere il gioco;
* Visualizzazione classifica dei primi dieci giocatori e salvataggio di questa su di un file;
* Titoli di coda;
* Ringraziamenti finali con una semplice animazione;
* Suoni per il menù,svolgimento del gioco, colpo mancato, nave colpita, nave affondata e titoli di coda.

## 

## Analisi tecnica

int main():

* scopo: contenere le altre funzioni e gestire i cicli che regolano i turni e le varie fasi di gioco;

Suoni menuIniziale(Suoni impost):

* scopo: spiegare il regolamento e consentire la rimozione delle musiche e/o degli effetti speciali.

void inputGiocatori(Statistiche gioc[], Statistiche vincitori[]):

* scopo: far inserire i nomi dei giocatori, controllando che non siano duplicati.

void coutGriglia(Statistiche gioc[], int turno, Dim lung, int griglia[][20][20], int caso):

* scopo: Mostrare graficamente la griglia di gioco;
* casi possibili: 1 griglia, per il posizionamento delle navi, 1 griglia e 1 radar per l’attacco, 2 griglie e 2 radar per la fine delle partite.

void cambioTurno(Statistiche gioc[], int p):

* scopo: interrompere il flusso del programma dopo ogni turno per evitare che un giocatore veda la griglia dell’altro, in modo da potersi scambiare comodamente.

void posizNavi(Dim lung, int mat[][20][20], Statistiche gioc[], int p):

* scopo: far posizionare le navi sulla griglia di gioco, eventualmente modificandole, con controllo che due navi non si sovrappongano.

Dim cinGriglia():

* scopo: far scegliere al giocatore la dimensione della griglia.

void coutElementi (int turno, int griglia[][20][20], Dim lung, int caso):

* scopo: mostrare le posizioni delle navi e dell’acqua, con colori diversi in caso la cella siia stata colpita da uno sparo. Associata alla funzione coutGriglia.

void gestNavi (Statistiche gioc[], Dim lung):

* scopo: far scegliere al giocatore un elenco di navi da uno dei file predefiniti o fargliene creare uno, con opportuni controlli per evitare che la dimensione delle celle totali non sia troppo grande e che le navi inserite non siano troppe.

void creaFile():

* scopo: creare i file che contengono gli elenchi predefiniti di navi.

Dim attacco(Dim lung, int mat[][20][20], int p, Statistiche gioc[], Suoni impost, Dim colpo):

* scopo: far scegliere al giocatore di turno le coordinate della cella da attaccare.

bool checkWin(Statistiche gioc[], Statistiche vincitori[], int t):

* scopo: controllare se la partita è finita e in tal caso aggiungere il vincitore alla classifica, per poi riordinarla e salvarla su un apposito file.

void inizMat(int griglia[][20][20]):

* scopo: inizializzare la matrice, in modo da evitare bug legati alle celle.

void coda(Suoni impost):

* scopo: mostrare i titoli di coda.

int verificaColpo(Statistiche gioc[], int p, Dim colpo, Suoni impost);

* scopo: controllare se il colpo è andato a vuoto, se ha colpito una nave o se la ha affondata.

void schermFin();

* scopo: mostrare una schermata finale con i ringraziamenti.

int coutLog(Dim lung, Dim colpo, Statistiche gioc[], int p, int stato, char log[][20][20], int n, bool isAttacco);

* scopo: mostrare una lista deglie eventi più recenti relativi agli attacchi effettuati;

void nave();

* scopo: mostrare una semplice animazione nella schermata finale.

## Manuale per l’utente

### Obiettivo

Divertirsi con una versione digitale della battaglia navale con gli amici;

### Avvio

Basterà avviare codeblocks e cliccare su build e run, o aprire il file di gioco dalla shell. In seguito sarete guidati passo per passo da un'interfaccia molto facile ma concisa che vi spiegherà come continuare.

### 

### Screenshot del programma in esecuzione

### 

### Flusso del programma

Dopo aver avviato il gioco, è necessario scegliere i nomi dei giocatori, le dimensioni della griglia e le navi da utilizzare. A turno i giocatori dovranno posizionare le loro navi, specificando la riga e la colonna, oltre all’orientamento. Nella successiva fase d’attacco, si dovranno invece indicare le coordinate della cella da colpire. Si otterranno dei resoconti sonoro e visivi dell’esito dell’attacco. Questa fase si ripete fino a quando uno dei due giocatori non ha affondato tutte le navi dell’altro. Dopo aver visualizzato le quattro griglie insieme, si potrà decidere di fare un’altra partita o procedere con la visualizzazione della classifica.

### Requisiti minimi

* Sistema operativo: Windows
* 1 MB RAM
* Memoria: 15 MB di spazio disponibile

## Test data set\Debug

I bug principali riscontrati durante lo sviluppo sono dovuti a un OBOE (Off By One Error), in particolare le posizioni delle navi venivano salvate nella matrice a partire dalla cella 1 1 anziché dalla 0 0, causando problemi in molteplici aspetti.

Fra i bug ancora riscontrabili c’è la possibilità che, a causa di problemi con il buffer, nella lista degli eventi due scritte si sorappongano.

## Bibliografia\Sitografia di riferimento

<https://www.youtube.com/watch?v=sbn-7WwXMvo> (tutorial per riprodurre un suono in C++)

Corso di Informatica di Fiorenzo Formichi e Giorgio Meini Zanichelli editori

## Osservazioni e conclusioni personali

La realizzazione di questo programma è stata utile sotto diversi aspetti, come il miglioramento della propria abilità nella programmazione e delle capacità di organizzazione e di lavorare in gruppo. E’ stato inoltre formativo imparare a riconoscere e risolvere alcuni tipi di bug, che non mancano durante la realizzazione di programmi complessi quanto o più di questo.