Battleship

v. 1.2

**done by Heorhii Sahliani**

**(group B22, Vinnytsia IT-Academy)**

**Task obtained: 22.12.2022**

**Pre-defense: \_\_.01.2023, Teacher 1\_\_\_\_\_\_\_\_\_\_\_**

**Defense: \_\_.01.2023, Teacher 2\_\_\_\_\_\_\_\_\_\_\_**

**CONTENTS**

**1. System requirements**

**2. About game “Battleship” (typical rules)**

**3. User guide**

**4. Programmer instruction**

**5. About the author**

**1. System requirements**

**Operating system:** Windows 10 64-bit.

**Hardware requirements:** user (programmer) requires

- processor: Pentium Processor;

- RAM: 640 Kb (min);

- Hard Disc: 1 Gb (min).

**2. About the “Battleship” game (typical rules)**

Battleship is a classic board game for two players, usually played on a grid. Each player has a fleet of ships, which include 1 Carrier, 2 Battleships, 3 Cruisers, 4 Patrol boats. The players take turns guessing the coordinates of the other player's ships with the goal of sinking all of their opponent's ships before they sink all of yours.

Here are the basic rules of the game:

1. Each player sets up their own fleet of ships on their own grid.
2. Players take turns guessing coordinates on the opponent's grid by calling out a letter and number combination (such as "A3").
3. If the opponent has a ship located at that spot, they will say "hit" and the player gets to take another turn.
4. If there is no ship at that spot, the opponent will say "miss" and the turn passes to the opponent.
5. When a player hits all the coordinates of an opponent's ship, that ship is considered sunk, and the opponent must announce which ship has been sunk.
6. The game continues until one player sinks all of the opponent's ships or runs out of turns.
7. The player who sinks all of their opponent's ships first wins.

The game can be played on a physical board, using a computer or mobile device, or even with pen and paper. The game can be played with different variations and levels of difficulty, with different board sizes, and different types of ships.

**3. User guide**

1. Find and run file Battleship.exe
2. Game screen shows below:

**-** a board with your ships on the left and a board with enemy ships on the right, both maps have coordinates along their sides.

**-** a status board showing how many ships you and your enemy have left, on the right.

- game messages to the player, at the bottom.

**c.** Input the coordinates “ x y ” of a square you want to shoot at (for example: “4 2”) and press Enter. If you input incorrect coordinates, the program will ask you to repeat this step.

**d.** You will take turns with a computer, until one loses all of their ships. Then the game will end and you will see a win or lose message depending on the result.

**e.** After the game ends press any key to restart the game or close the window to quit.

**4. Programmer instruction**

The project was written on C++.

Advantages of C++:

**Portability:** C++ is one of the most used and portable to different platforms, almost any type of computer and operating system.

**Brevity: c**ode written in C++ is very short in comparison with other languages.

**Modular programming:** applications written on C++ can be made up of several source code files which are compiled separately and then linked together. This characteristic allows to link C++ code with the ones written in other languages (for example, Assembler).

**Speed:** The resulting code from a C++ compilation is very efficient due to the reduced size of the language itself.

Project is a single CppUnit which contains all necessary algorithms and functions.

Find and open the file Battleship.cpp. For edition, debugging and compiling program code you may use C++ compilers such as MS Visual Studio, Visual C++, C++ Builder, Borland C++, Dev C++ and similar.

Main global variables:

const int N = 10 – amount of ships to be placed on the map.

const int NUM\_SHIPS = 10 – amount of ship indexes.

Main functions:

void cursorxy () - sets console cursor position according to coordinates x, y.

bool ship\_in\_map () – checking if every ship is within the borders of the map.

void place\_rand\_ships – placing ships on the map if placement is possible.

void show\_ships\_status – outputs a status board that shows a current amount of alive ships on the map

void map\_show – outputs a map which corresponds to the current game progress.

int shoot - processing the entered coordinates (x, y) and shooting at the corresponding cell on the map

**5. About the author**

This project was done by Heorhii Sahliani – Vinnitsya IT-Academy student (group B22, January 2023).

Special thanks to my friend Andrii Dobrovolskyi, who tested the game with me.