**Warship Game – Work Plan**

**Project start date:** August.

**Introduction:**

Warship Game is the old classic turns game in which the players try to sink the opponent's navy. First, each player secretly places their ships on the boards. Then, in each turn, the player chooses where to shell in the opponent's board. After the player confirms the attack, the BDA (Bombing Damage Assessment) of the attack is shown, which can be an ALFA (a rival's ship was hit) or a MISS. The player can also watch his own ships' status as well. The first player that destroys the opponent's navy wins.

This Visual Studio version is coded in C and played on the black console. The game starts with an introduction to the roles, and the players insert their names. Like in the classic game, each player places his ships on the board, each ship is placed in a different coordinate, horizontally or vertically. The first player to start is chosen randomly. In each turn, the players shell each player's navy until the first one to be destroyed. At that point, the game declares the winner and ends.

**Code plan:**

Types of ships:

|  |  |  |
| --- | --- | --- |
| **Type** | **Presented as** | **Amount in the game** |
| Mother Ship | <MMM> | 1 |
| Submarine | <SU> | 2 |
| Warship | <W> | 3 |
| Combat Ship | <> | 4 |

The board:

The board size is 10X10, and the rows and columns numbers are presented for easier orientation. The board's symbols:

|  |  |  |
| --- | --- | --- |
| **Type** | **Presented as** | **Color** |
| Sea | ~ | Blue |
| Ship | < | White |
| A hit | X | Red |
| A Miss | O | White |

An example of a board:

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| X|Y | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
| 1 | ~ | ~ | ~ | < | M | M | X | X | ~ | ~ |
| 2 | ~ | ~ | < | S | U | > | ~ | X | > | ~ |
| 3 | ^ | < | > | ~ | ~ | ~ | ~ | ~ | ~ | ^ |
| 4 | S | ~ | ~ | ~ | ~ | ~ | O | O | ~ | W |
| 5 | U | ~ | ~ | ~ | ~ | ~ | ~ | O | ~ | V |
| 6 | V | ~ | ~ | ~ | < | W | > | ~ | ~ | ~ |
| 7 | ~ | ~ | O | ~ | < | > | ~ | ~ | ~ | ^ |
| 8 | < | > | ~ | ~ | ~ | ~ | ~ | ~ | ~ | W |
| 9 | ~ | ~ | ~ | < | X | ~ | ~ | ~ | ~ | V |
| 10 | ~ | ~ | ~ | ~ | ~ | ~ | ~ | ~ | ~ | ~ |

Structures:

* Point – contains two coordinates: X and Y.
* Ship – contains the type of the ship, the size, the direction (horizontal or vertical), and a pointer to the first up or left coordinate of the ship.
* Board – contains a matrix of characters that represent the objects of the game (ship, sea, hit, or miss), and an array of the ships.
* Player – contains the number of the player, name, the player's board (blue board), and the unknown opponent's board (red board).

Functions:

* Initiate functions:
  + Red board initiation – matrix dynamic memory allocation.
  + Blue board initiation – matrix and ships dynamic memory allocation.
  + Player initiation – name, red board, blue board dynamic memory allocation.
* Check functions:
  + Ship position validation check – checks that the ship position that was chosen by the player doesn’t clash with other ships on the board, or exceed the board limits.
  + Shoot validation check.
  + Check if the player is still alive.
* Game functions:
  + Board printing.
  + Ships placing.
  + Instructions printing.
  + Dice rolling (to decide who starts).
  + Player turn – managing the player's turn.
  + Attacking.
* Deallocate memory functions:
  + Red board deallocation.
  + Blue board deallocation.
  + Player deallocation.
* Colors functions:
  + Blue, red, and reset.
* Game-on function – a function that operates all the necessary functions for running the game.

Constants:

The constants for the code are: Player's name maximum length, board size, number of ships, the size of each ship, number of shoots for a player.