Minesweeper Game

Author: Piotr Copek

Short description of the game

> Single-player puzzle game which requires players to eliminate mines on a grid without tripping any. The field for the game is a square grid which contains mines in some of its squares. These squares are uncovered by the player and if one happens to contain a mine then the game ends immediately. If a square does not have a mine, it reveals a number indicating how many neighboring squares have them. The numbers are used by the player to identify where the bombs may be through marking them. The aim of winning this game is to uncover all squares without bombs inside them.

Classes Overview

Cell

The `Cell` class is used to represent a single cell on a Minesweeper grid.

Properties

- `is_bomb` – a boolean flag that indicates if the cell is a bomb.
- `is_marked` – a boolean flag used to indicate if a cell has been flagged by the player as a bomb.
- `is_revealed` – a boolean flag that shows if the cell has already been revealed.
- `bombs_around` – an integer counter for the number of bombs placed around the cell.

Methods

- `mark_as_bomb()` – toggles the `is_bomb` state if it hasn't been toggled already.
- `toggle_mark_flag()` – toggles the `is_marked` flag.

- `reveal()` – sets the `is_revealed` flag to true.
- `update_counter()` – adds 1 to the `bombs_around` counter.

Board

The `Board` class is responsible for managing the entire Minesweeper game.

Properties

- `board` – 2D vector of `Cell` objects.
- `width` – the width of the board.
- `height` – the height of the board.
- `bomb_amount` – the number of bombs on the board.
- `first_x`, `first_y` – coordinates used to store the first move played by the player.

Methods

- `create_board()` – initializes the board with `Cell` instances and sets up the graphical user interface.
- `count_bombs(int x, int y)` – counts the number of bombs around a cell.
- `random_bomb_placement()` – randomly places bombs on the board, ensuring that the first move is always safe.
- `get_user_move()` – gathers the user input including row, column, and mode.
- `verify_input()` – checks the validity of user input.
- `update_board()` – executes the user's move and updates the state of the game board.
- `handle_first_mode(Board &board, Cell &cell, int row, int column)` – handles the user's action to reveal a cell.
- `show_around(int row, int column)` – shows the cells around a given cell, recursively searching for empty cells.
- `handle_second_mode(Cell &cell)` – handles marking and un-marking a cell as a bomb.
- `handle_win_conditions()` – verifies if the player has correctly marked all bombs and won the game.
- `handle_lose_condition()` – reveals all cells on the board when the game is lost.

GUI

The `GUI` class is responsible for displaying the current state of the board to the terminal.

Properties

- `board` – a 2D vector of `Cell` objects.
- `fancy_print` – a copy of `board`.

Methods

- `print_board(const Board &board)` – prints the `fancy_print` to the console.
- `clear_terminal()` – clears the screen of the console.
- `welcome_screen()` – displays the welcome message.
- `end_screen()` – displays the end message.

Player

The `Player` class is responsible for handling player interactions and input.

Methods

- `player_move(int board_x, int board_y)` – prompts the player to input the row, column, and mode of their move. Returns a tuple of `(row, column, mode)`.
- `verify_input(int &input, const std::string &prompt, int min, int max)` – prompts the user with a message to input a value. Checks if the input is a valid integer within the specified range. Returns true if the input is valid; otherwise, it prompts again until a valid input is provided.
- `get_board_dimensions()` – prompts the player to input the width and height of the game board. Returns a tuple of `(width, height)`.
- `play_again()` – retrieves if the player wants to play again.

Randomizer

The `Randomizer` class provides methods for generating random numbers within specific range.

Properties

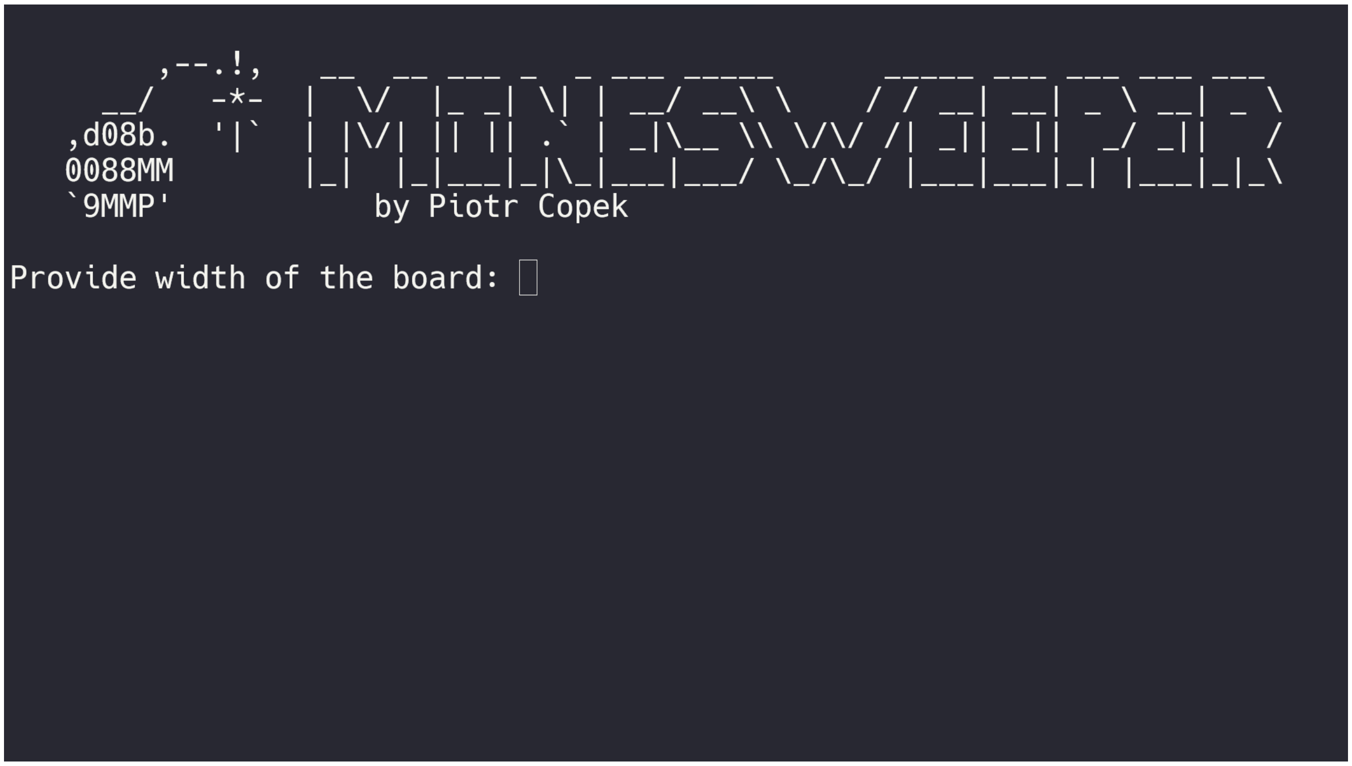
- `gen` – a Mersenne Twister random number generator initialized with a seed based on the current time.

Methods

- `RandomNumberGenerator()` – constructor that initializes the random number generator with a seed based on the current time.
- `get_random_number(int min, int max)` – generates and returns a random integer in the given range.

Project showcase

Welcome screen

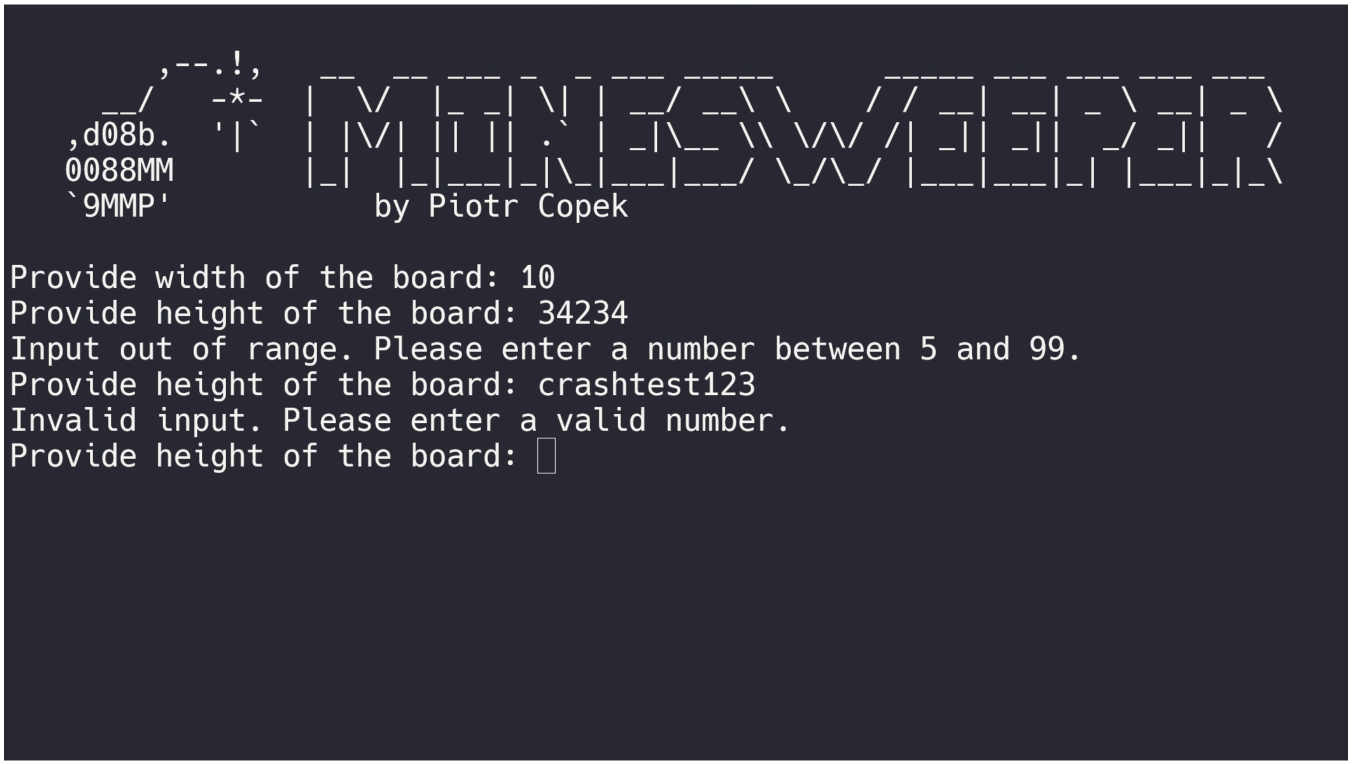


```
        ,--.!,   __  __ ___ _  _ ___ _____      _____ ___ ___ ___ ___
     __/   -*-  |  \/  |_ _| \| | __/ __\ \    / / __| __| _ \ __| _ \
   ,d08b.  '|`  | |\/| || || .` | _|\__ \\ \/\/ /| _|| _||  _/ _||   /
   0088MM       |_|  |_|___|_|\_|___|___/ \_/\_/ |___|___|_| |___|_|_\
   `9MMP'           by Piotr Copek

Provide width of the board: 
```

Dimensions of the board

Only integers in range 5 to 50 are accepted. Incorrect input won`t be accepted and program will prompt user to enter correct value.

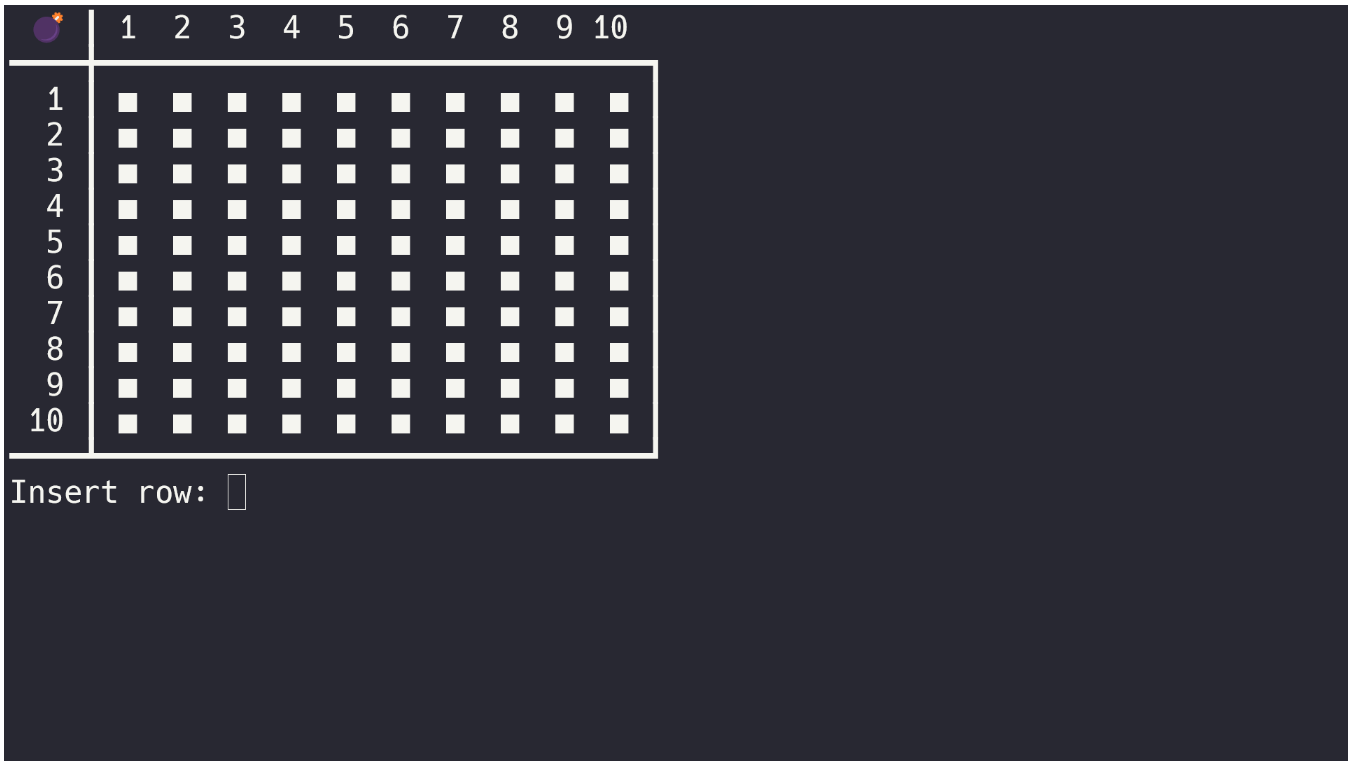


```
        ,--.!,   __  __ ___ _  _ ___ _____      _____ ___ ___ ___ ___
     __/   -*-  |  \/  |_ _| \| | __/ __\ \    / / __| __| _ \ __| _ \
   ,d08b.  '|`  | |\/| || || .` | _|\__ \\ \/\/ /| _|| _||  _/ _||   /
   0088MM       |_|  |_|___|_|\_|___|___/ \_/\_/ |___|___|_| |___|_|_\
   `9MMP'           by Piotr Copek

Provide width of the board: 10
Provide height of the board: 34234
Input out of range. Please enter a number between 5 and 99.
Provide height of the board: crashtest123
Invalid input. Please enter a valid number.
Provide height of the board: 
```

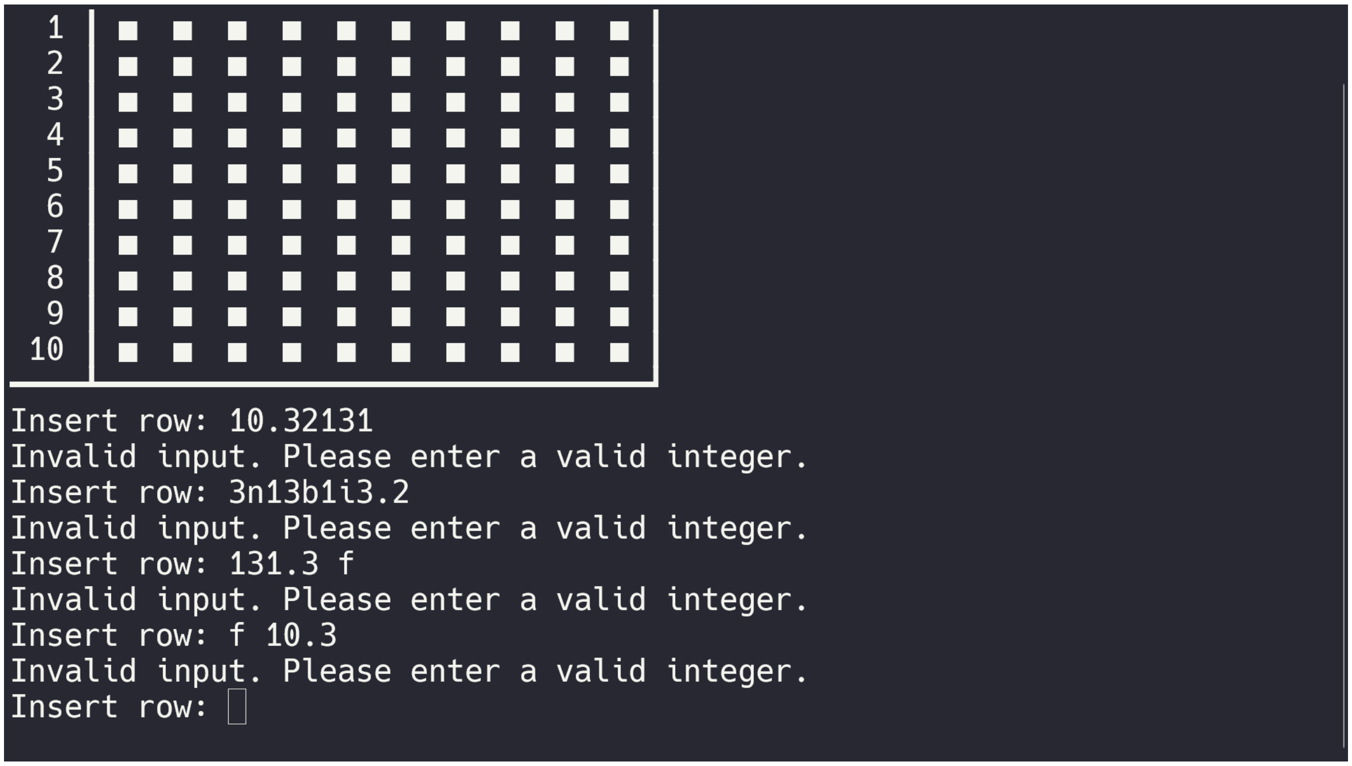
Game

After entering valid input the game will start. The empty board will be printed and user will be asked for first move.



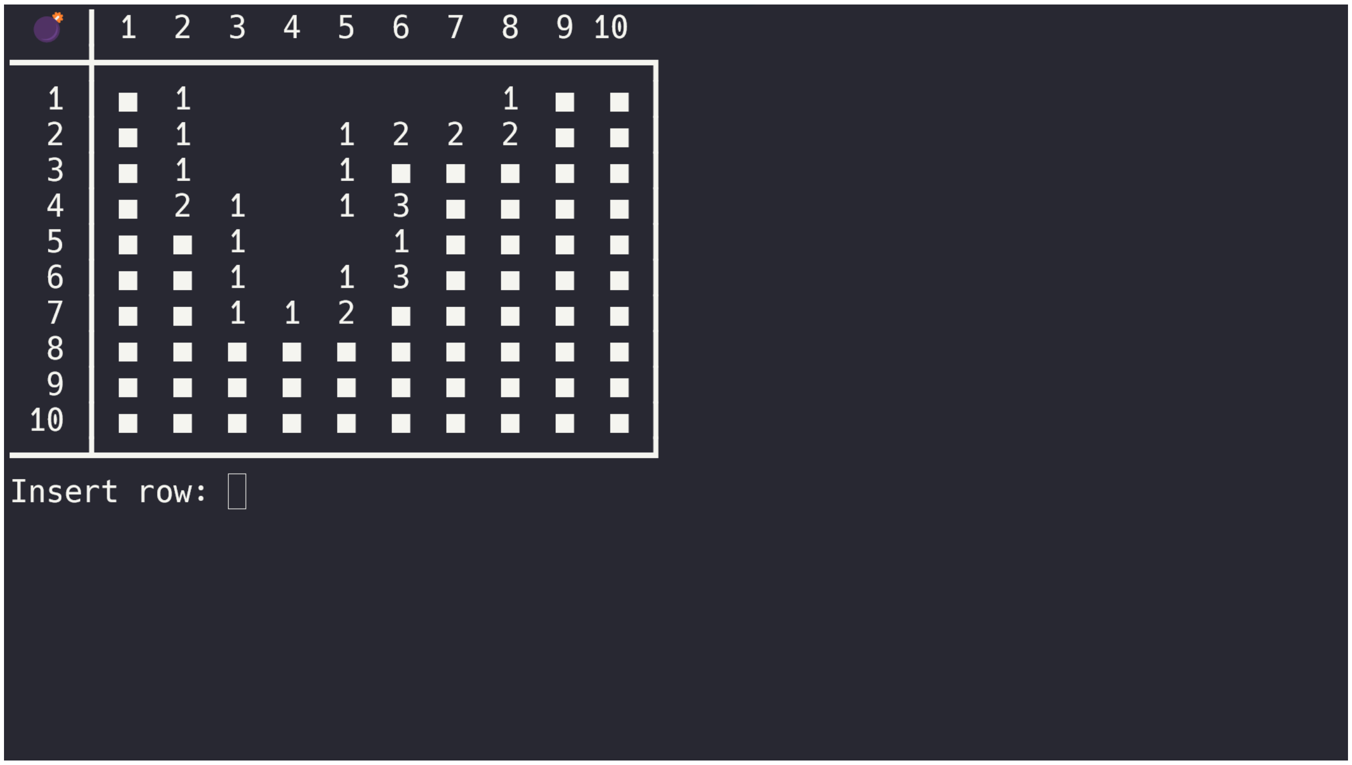
```
 💣 │ 1  2  3  4  5  6  7  8  9 10
────┼──────────────────────────────┐
  1 │ ■  ■  ■  ■  ■  ■  ■  ■  ■  ■ │
  2 │ ■  ■  ■  ■  ■  ■  ■  ■  ■  ■ │
  3 │ ■  ■  ■  ■  ■  ■  ■  ■  ■  ■ │
  4 │ ■  ■  ■  ■  ■  ■  ■  ■  ■  ■ │
  5 │ ■  ■  ■  ■  ■  ■  ■  ■  ■  ■ │
  6 │ ■  ■  ■  ■  ■  ■  ■  ■  ■  ■ │
  7 │ ■  ■  ■  ■  ■  ■  ■  ■  ■  ■ │
  8 │ ■  ■  ■  ■  ■  ■  ■  ■  ■  ■ │
  9 │ ■  ■  ■  ■  ■  ■  ■  ■  ■  ■ │
 10 │ ■  ■  ■  ■  ■  ■  ■  ■  ■  ■ │
────┴──────────────────────────────┘
Insert row: 
```

If invalid string is inserted the user will be prompted again. User can`t input float, double, letters and special characters.



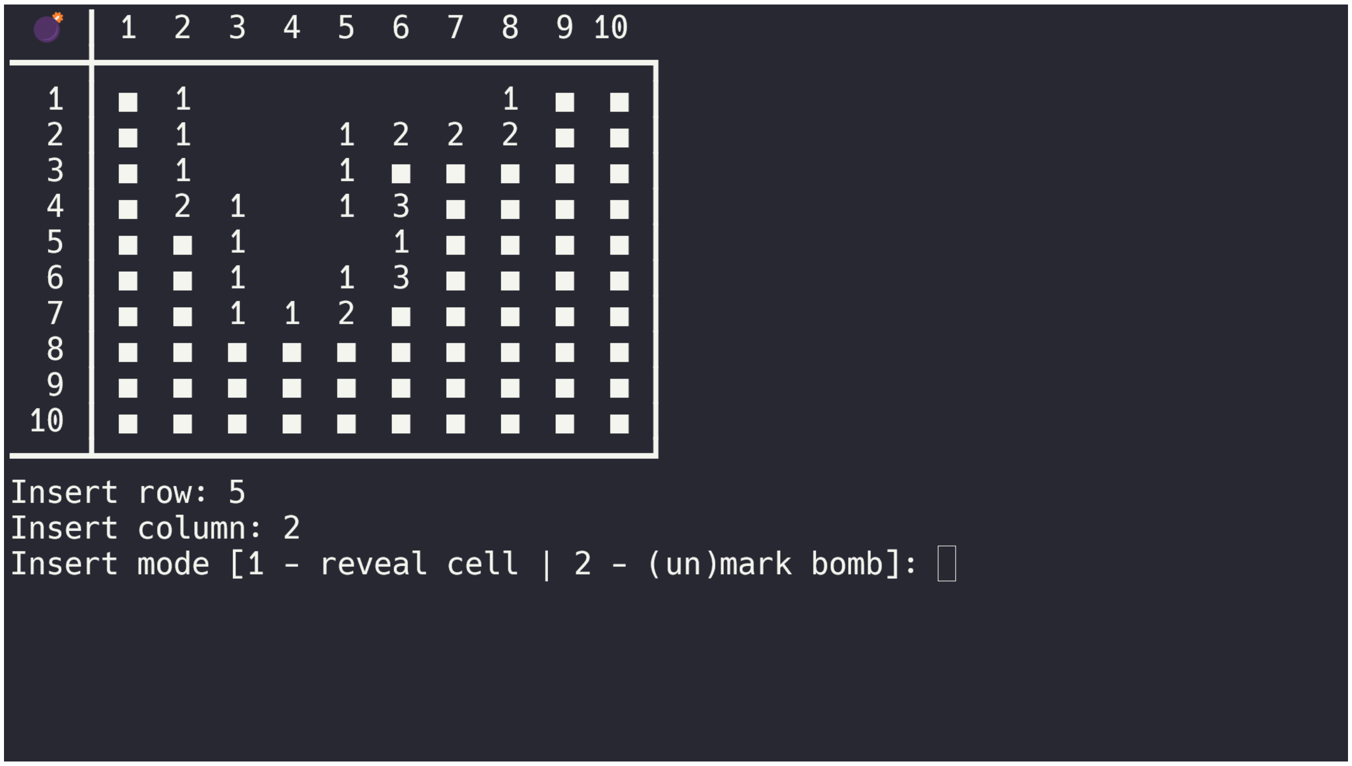
```
  1 ┃ ■  ■  ■  ■  ■  ■  ■  ■  ■  ■ ┃
  2 ┃ ■  ■  ■  ■  ■  ■  ■  ■  ■  ■ ┃
  3 ┃ ■  ■  ■  ■  ■  ■  ■  ■  ■  ■ ┃
  4 ┃ ■  ■  ■  ■  ■  ■  ■  ■  ■  ■ ┃
  5 ┃ ■  ■  ■  ■  ■  ■  ■  ■  ■  ■ ┃
  6 ┃ ■  ■  ■  ■  ■  ■  ■  ■  ■  ■ ┃
  7 ┃ ■  ■  ■  ■  ■  ■  ■  ■  ■  ■ ┃
  8 ┃ ■  ■  ■  ■  ■  ■  ■  ■  ■  ■ ┃
  9 ┃ ■  ■  ■  ■  ■  ■  ■  ■  ■  ■ ┃
 10 ┃ ■  ■  ■  ■  ■  ■  ■  ■  ■  ■ ┃
━━━━┻━━━━━━━━━━━━━━━━━━━━━━━━━━━━━┛
Insert row: 10.32131
Invalid input. Please enter a valid integer.
Insert row: 3n13b1i3.2
Invalid input. Please enter a valid integer.
Insert row: 131.3 f
Invalid input. Please enter a valid integer.
Insert row: f 10.3
Invalid input. Please enter a valid integer.
Insert row: 
```

After entering correct integers the game will begin. The bombs will be placed randomly on board also ensuring the bomb won`t lay on cell which has been chosen as the first move. Example of the board after entering 4th row and 4th column:



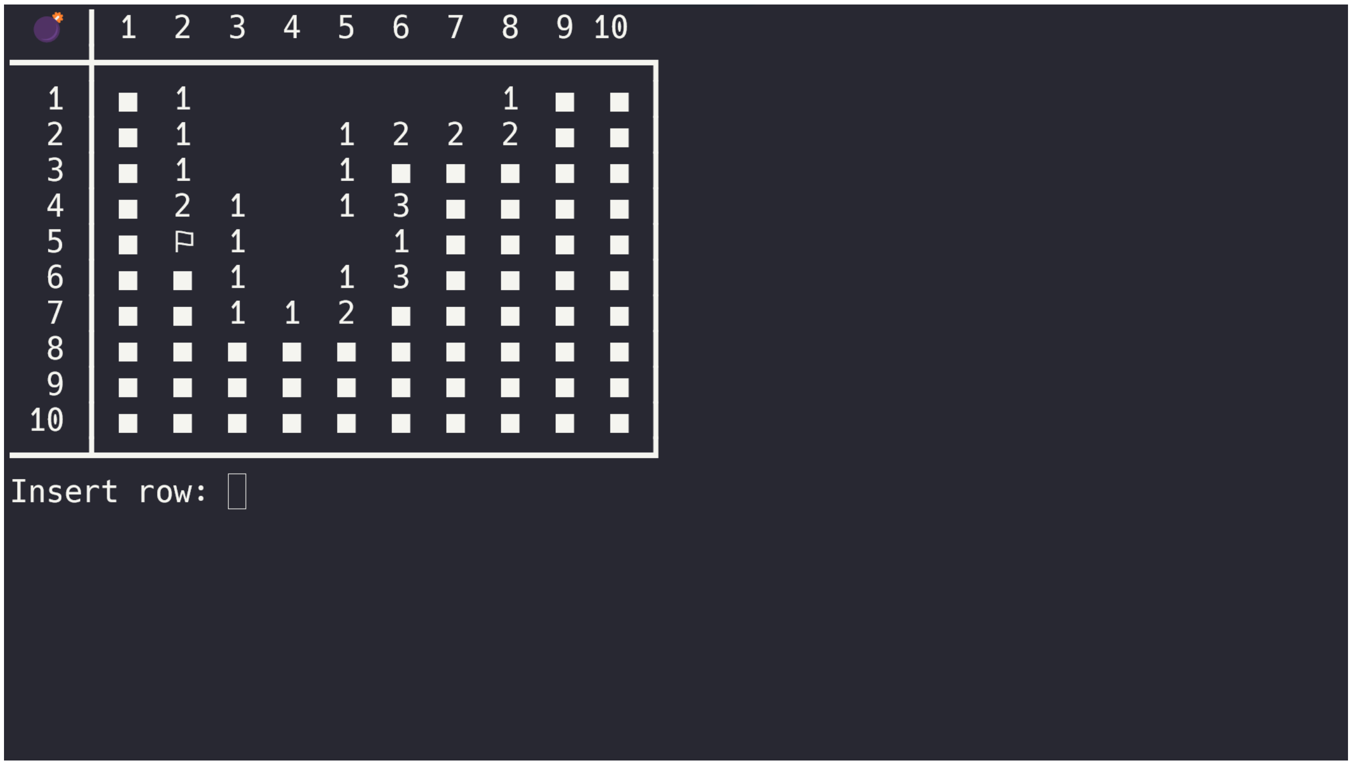
```
 💣 ┃ 1  2  3  4  5  6  7  8  9 10
━━━━╋━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━┓
  1 ┃ ■  1                 1  ■  ■ ┃
  2 ┃ ■  1        1  2  2  2  ■  ■ ┃
  3 ┃ ■  1        1  ■  ■  ■  ■  ■ ┃
  4 ┃ ■  2  1     1  3  ■  ■  ■  ■ ┃
  5 ┃ ■  ■  1        1  ■  ■  ■  ■ ┃
  6 ┃ ■  ■  1     1  3  ■  ■  ■  ■ ┃
  7 ┃ ■  ■  1  1  2  ■  ■  ■  ■  ■ ┃
  8 ┃ ■  ■  ■  ■  ■  ■  ■  ■  ■  ■ ┃
  9 ┃ ■  ■  ■  ■  ■  ■  ■  ■  ■  ■ ┃
 10 ┃ ■  ■  ■  ■  ■  ■  ■  ■  ■  ■ ┃
━━━━┻━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━┛
Insert row: 
```

Next moves have two variants - revealing cell or (un)marking bomb.



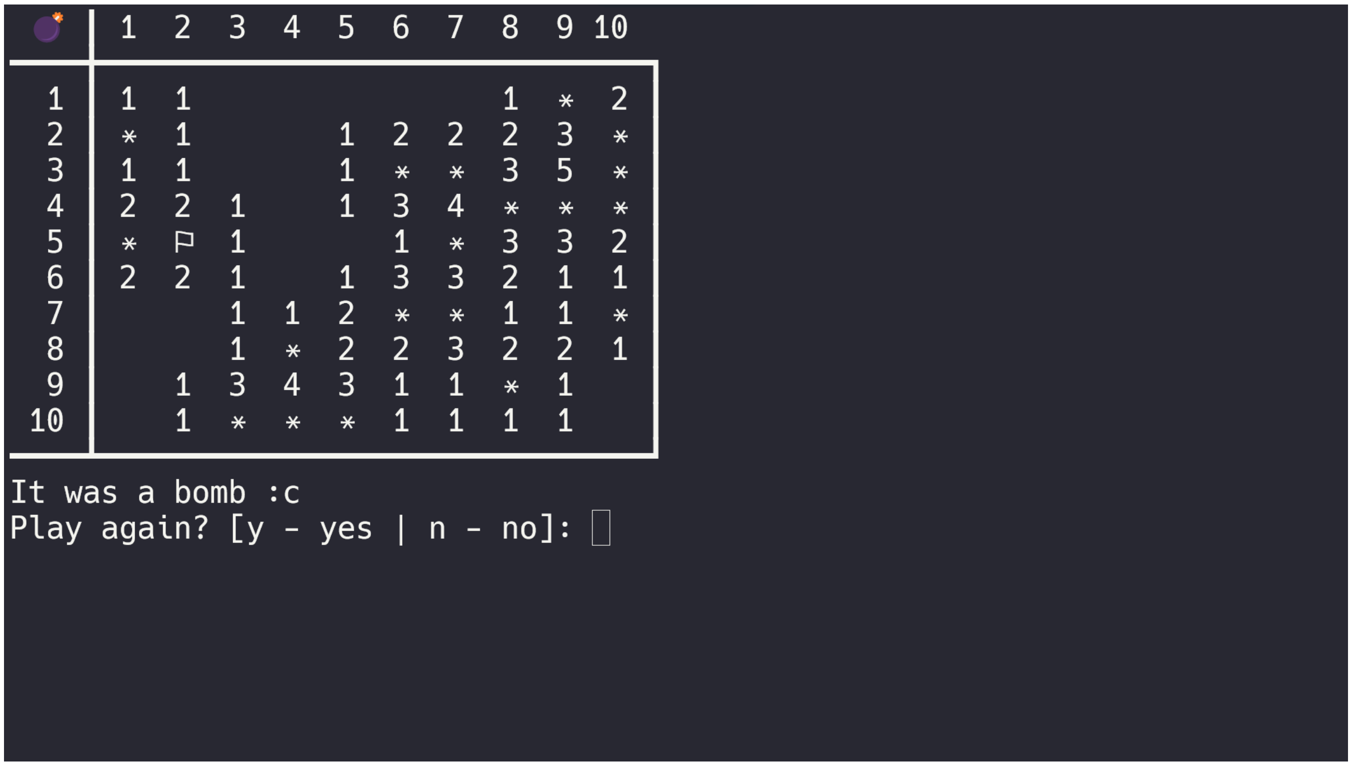
```
 💣 ┃ 1  2  3  4  5  6  7  8  9 10
━━━━╋━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━┓
  1 ┃ ■  1                 1  ■  ■ ┃
  2 ┃ ■  1        1  2  2  2  ■  ■ ┃
  3 ┃ ■  1        1  ■  ■  ■  ■  ■ ┃
  4 ┃ ■  2  1     1  3  ■  ■  ■  ■ ┃
  5 ┃ ■  ■  1        1  ■  ■  ■  ■ ┃
  6 ┃ ■  ■  1     1  3  ■  ■  ■  ■ ┃
  7 ┃ ■  ■  1  1  2  ■  ■  ■  ■  ■ ┃
  8 ┃ ■  ■  ■  ■  ■  ■  ■  ■  ■  ■ ┃
  9 ┃ ■  ■  ■  ■  ■  ■  ■  ■  ■  ■ ┃
 10 ┃ ■  ■  ■  ■  ■  ■  ■  ■  ■  ■ ┃
━━━━┻━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━┛
Insert row: 5
Insert column: 2
Insert mode [1 - reveal cell | 2 - (un)mark bomb]: 
```

If we choose 2nd option, flag will appear on the selected cell.



```
💣 │ 1  2  3  4  5  6  7  8  9 10
───┼──────────────────────────────┐
 1 │ ■  1                 1  ■  ■ │
 2 │ ■  1        1  2  2  2  ■  ■ │
 3 │ ■  1        1  ■  ■  ■  ■  ■ │
 4 │ ■  2  1     1  3  ■  ■  ■  ■ │
 5 │ ■  ⚐  1        1  ■  ■  ■  ■ │
 6 │ ■  ■  1     1  3  ■  ■  ■  ■ │
 7 │ ■  ■  1  1  2  ■  ■  ■  ■  ■ │
 8 │ ■  ■  ■  ■  ■  ■  ■  ■  ■  ■ │
 9 │ ■  ■  ■  ■  ■  ■  ■  ■  ■  ■ │
10 │ ■  ■  ■  ■  ■  ■  ■  ■  ■  ■ │
───┴──────────────────────────────┘
Insert row:
```

After trying to reveal cell on which bomb is placed the game will end, showing the fully revealed board and asking for new game.



```
💣 │ 1  2  3  4  5  6  7  8  9 10
───┼──────────────────────────────┐
 1 │ 1  1                 1  *  2 │
 2 │ *  1        1  2  2  2  3  * │
 3 │ 1  1        1  *  *  3  5  * │
 4 │ 2  2  1     1  3  4  *  *  * │
 5 │ *  ⚐  1        1  *  3  3  2 │
 6 │ 2  2  1     1  3  3  2  1  1 │
 7 │       1  1  2  *  *  1  1  * │
 8 │       1  *  2  2  3  2  2  1 │
 9 │    1  3  4  3  1  1  *  1    │
10 │    1  *  *  *  1  1  1  1    │
───┴──────────────────────────────┘
It was a bomb :c
Play again? [y - yes | n - no]:
```

After selecting "n" as an option the game will exit with ascii art of bomb, otherwise program will ask for new board dimensions and start new game.

```
        ,--.!,
     __/   -*-
   ,d08b.  '|`
   0088MM
   `9MMP'

┏[piotr from LAPTOP-8JKTK9N3][9:49.414s][RAM: 10/15GB][Saturda
y at 10:53:09 PM][main ≡]
┗[~\Documents\Studies\Sem_02\CP2\labs\a41c0462-gr11-repo\Project]
└─Δ
```