

Wordle Game and Solver by AZZOUZ

Abdenour

Running the Program

Execute the program:

```
./wordle
```

Or use:

```
make run
```

Usage

When you run the program, you'll see a menu:

```
==== WORDLE MENU ====
1. Play Game
2. Run Solver
3. Exit
Choice:
```

Option 1: Play Game

- You'll be prompted to guess a 5-letter word
- You have 6 attempts to guess correctly
- Feedback format:
 - [x] = correct letter in correct position (green)
 - (x) = correct letter in wrong position (yellow)
 - x = letter not in word (gray)

Option 2: Run Solver

- Watch the automated solver attempt to guess the target word
- See the strategy in action as it narrows down possibilities

Example Output

Playing the Game:

```
==== WORDLE GAME ====
Guess the 5-letter word! You have 6 tries.
[X] = correct, (X) = wrong position, X = not in word

Attempt 1/6: crane
Guess: c   r (a) n   e

Attempt 2/6: about
Guess: [a] b   o   u   t

Attempt 3/6: alarm
Guess: [a][l][a][r][m]

Congratulations! You won in 3 attempts!
```

Running the Solver:

```
==== WORDLE SOLVER ====
Target word: train (hidden)

Attempt 1: Guess: s   o (a) r   e
Remaining possibilities: 45

Attempt 2: Guess: (t)(r)(a) i   n
Remaining possibilities: 8

Attempt 3: Guess: [t][r][a][i][n]

Solver found the word in 3 attempts!
```

Files

- `main.c` - Main program with menu interface

- `game.c/h` - Wordle game logic
- `solver.c/h` - Automated solver implementation
- `utils.c/h` - Utility functions (dictionary loading, feedback display)
- `words.txt` - Dictionary of valid 5-letter words
- `Makefile` - Build configuration

Algorithm

The solver uses:

1. **First guess:** "soare" (high frequency letters)
2. **Filtering:** Eliminates words that don't match feedback
3. **Next guess selection:** Chooses word with highest letter frequency score among remaining possibilities

Requirements

- GCC compiler
- Standard C library
- Unix-like environment (Linux/Mac) or Windows with MinGW