**1. Podręcznik użytkownika programu**

**Opis zasad gry**

Minesweeper to klasyczna gra polegająca na odkrywaniu pól na planszy w celu uniknięcia min. Celem gry jest odkrycie wszystkich pól, które nie są minami, bez detonowania żadnej miny. Na każdym z odkrytych pól znajduje się liczba min bezpośrednio stykających się z danym polem (od zera do ośmiu). Gracz może oznaczyć pole flagą, co chroni je przed przypadkowym odsłonięciem.

**Sposób uruchomienia programu**Aby uruchomić program, należy najpierw go skompilować, a następnie wykonać wygenerowany plik wykonywalny.

1. **Kompilacja:**

|  |
| --- |
| 1 cd src  2 make |

1. **Uruchomienie gry:**

|  |
| --- |
| ./minesweeper |

**Opis poszczególnych opcji**

* **Wybór poziomu trudności:**
  + Po uruchomieniu programu, użytkownik jest proszony o wybór poziomu trudności:
    - **1 - Easy** (9x9 pól, 10 min)
    - **2 - Medium** (16x16 pól, 40 min)
    - **3 - Hard** (16x30 pól, 99 min)
    - **4 - Custom** (użytkownik wprowadza własne wymiary planszy i liczbę min)
* **Wprowadzanie komend:**
  + **f x y**: Ustawia lub usuwa flagę na polu o współrzędnych **(x, y)**.
  + **r x y**: Odsłania pole o współrzędnych **(x, y)**.
* **Uruchomienie z pliku:**
  + Aby uruchomić grę z pliku, należy podać ścieżkę do pliku jako argument:

|  |
| --- |
| ./minesweeper -f ../data/game.txt |

**Przykład pliku gry (game.txt)**

Plik **game.txt** powinien zawierać wymiary planszy, liczbę min i sekwencję komend do wykonania.

|  |
| --- |
| 6 6 6  1 1 1 0 0 0  3 \* 2 0 0 0  \* \* 2 0 1 1  3 3 1 \* 1 \*  \* 2 0 0 1 1  \* 2 0 0 0 0  r 0 0  f 1 1  r 1 2  r 2 2  f 3 3  r 4 4 |

**2. Szczegóły implementacji programu**

**Podział programu na moduły**

Program został podzielony na następujące moduły:

* **main.c** : Główny plik programu, obsługujący menu wyboru poziomu trudności i uruchamiający odpowiednią funkcję gry.
* **board.c i board.h** : Obsługa tworzenia, wyświetlania i manipulowania planszą gry.
* **game.c i game.h** : Logika gry, w tym uruchamianie gry, odczytywanie ruchów z pliku, sprawdzanie warunków zwycięstwa/przegranej i obliczanie punktów.
* **utils.c i utils.h** : Funkcje narzędziowe, takie jak zarządzanie wynikami, zapisywanie i odczytywanie wyników z pliku, sprawdzanie warunków zwycięstwa i obliczanie punktów.
* **tests.c** : Testy jednostkowe dla kluczowych funkcji programu.

**Interfejs kluczowych funkcji z opisem implementacji**

* **main.c**
  + **int main(int argc, char \*argv[])**: Główna funkcja programu, obsługująca wybór poziomu trudności i uruchamiającą grę.
* **board.c i board.h**
  + **Board\* create\_board(int rows, int cols, int mines)**: Tworzy nową planszę gry.
  + **void free\_board(Board \*board)**: Zwolnia pamięć zajmowaną przez planszę gry.
  + **void print\_board(Board \*board)**: Wyświetla planszę gry.
  + **void place\_mines(Board \*board, int first\_row, int first\_col)**: Losowo rozmieszcza miny na planszy, unikając pierwszego odsłoniętego pola.
  + **void reveal\_cell(Board \*board, int row, int col)**: Odsłania pole na planszy i odsłania również sąsiednie pola, jeśli odsłonięte pole ma wartość **0**.
  + **void toggle\_flag(Board \*board, int row, int col)**: Ustawia lub usuwa flagę na polu.
* **game.c i game.h**
  + **void run\_game(int difficulty)**: Uruchamia grę z wybranym poziomem trudności.
  + **void run\_game\_custom(int rows, int cols, int mines)**: Uruchamia grę z niestandardowymi wymiarami planszy i liczbą min.
  + **int run\_from\_file(const char \*filename)**: Uruchamia grę z pliku, odczytując wymiary planszy, liczbę min i sekwencję komend.
* **utils.c i utils.h**
  + **void read\_scores()**: Odczytuje i wyświetla top 5 wyników z pliku **scores.txt**.
  + **void save\_score(const char \*name, int score)**: Zapisuje wynik gracza do pliku **scores.txt**.
  + **int check\_win(Board \*board)**: Sprawdza, czy gracz wygrał, odkrywając wszystkie pola, które nie są minami.
  + **int calculate\_score(Board \*board, int difficulty)**: Oblicza wynik gracza na podstawie liczby odsłoniętych pól i poziomu trudności.
* **tests.c**
  + **void test\_create\_board()**: Testuje tworzenie i inicjalizację planszy.
  + **void test\_place\_mines()**: Testuje rozmieszczenie min na planszy.
  + **void test\_reveal\_cell()**: Testuje odsłanianie pojedynczego pola.
  + **void test\_toggle\_flag()**: Testuje ustawianie i usuwanie flagi na polu.
  + **void run\_tests()**: Uruchamia wszystkie testy.
  + **int main()**: Główna funkcja testów, uruchamiającą wszystkie testy jednostkowe.

**Opis ważniejszych struktur**

* **Board (zdefiniowane w board.h):**

|  |
| --- |
| typedef struct {  int rows;  int cols;  int mines;  int \*\*board;  int \*\*revealed;  int \*\*flags;  } Board; |

* + **rows**: Liczba wierszy na planszy.
  + **cols**: Liczba kolumn na planszy.
  + **mines**: Liczba min na planszy.
  + **board**: Dwuwymiarowa tablica przechowująca wartości pól (liczba min wokół pola lub **-1** dla miny).
  + **revealed**: Dwuwymiarowa tablica przechowująca informacje o odsłoniętych polach.
  + **flags**: Dwuwymiarowa tablica przechowująca informacje o oznaczonych flagami polach.

**Opis i wynik poszczególnych testów**

* **test\_create\_board()**
  + **Opis:** Tworzy planszę o rozmiarze 3x3 z 1 miną i sprawdza, czy plansza została poprawnie zainicjalizowana.
  + **Wynik:** Powinien wyświetlić **Test passed: create\_board**.
* **test\_place\_mines()**
  + **Opis:** Tworzy planszę o rozmiarze 3x3 z 1 miną i sprawdza, czy mina została poprawnie rozmieszczona, unikając pierwszego pola.
  + **Wynik:** Powinien wyświetlić **Test passed: place\_mines**.
* **test\_reveal\_cell()**
  + **Opis:** Tworzy planszę o rozmiarze 3x3 z 1 miną, odsłania pierwsze pole i sprawdza, czy pole zostało poprawnie odsłonięte.
  + **Wynik:** Powinien wyświetlić **Test passed: reveal\_cell**.
* **test\_toggle\_flag()**
  + **Opis:** Tworzy planszę o rozmiarze 3x3 z 1 miną, ustawia flagę na polu i sprawdza, czy flaga została poprawnie ustawiona i usunięta.
  + **Wynik:** Powinien wyświetlić **Test passed: toggle\_flag**.

**3. Podsumowanie**

**Czy wszystkie funkcjonalności udało się zaimplementować?**

Tak, wszystkie główne funkcjonalności zostały zaimplementowane:

* Wybór poziomu trudności.
* Tworzenie i wyświetlanie planszy.
* Rozmieszczenie min.
* Odsłanianie i oznaczanie pól flagą.
* Zarządzanie wynikami i zapisywanie wyników do pliku.
* Uruchamianie gry z pliku.

**Co sprawiało trudności?**

* **Obsługa wcięć w Makefile:** Pierwotnie używano spacji zamiast tabów, co powodowało błędy **missing separator**.
* **Dynamiczne wyrównanie nagłówków kolumn:** Wymiary planszy przekraczające 9 powodowały nieprawidłowe wyrównanie nagłówków kolumn.

**Jakie problemy napotkaliście i jak je rozwiązaliście?**

* **Problem z wcięciami w Makefile:**
  + **Rozwiązanie:** Użyto edytora **vim** do zamiany spacji na taby, co zapewniło poprawne wyrównanie wierszy zależnych.
* **Problem z dynamicznym wyrównaniem nagłówków kolumn:**
  + **Rozwiązanie:** Dodano obliczanie maksymalnej liczby cyfr w indeksach wierszy i kolumn oraz użycie formatowania **%\*d** w funkcji **print\_board** do dynamicznego wyrównania nagłówków kolumn.

**Państwa wnioski**

* **Modularność:** Podział programu na moduły ułatwia utrzymanie i rozszerzanie kodu.
* **Testowanie:** Implementacja testów jednostkowych pomaga w wykrywaniu błędów wczesno i zapewnia, że kluczowe funkcje działają poprawnie.
* **Dynamiczne wyrównanie:** Dynamiczne obliczanie maksymalnej liczby cyfr w indeksach wierszy i kolumn zapewnia prawidłowe wyrównanie planszy dla różnych rozmiarów.