| **Sprawozdanie** |
| --- |
| Autorzy: Magdalena Zegarowska, Szymon Opryński |
| Spis treści:<br>1. Opis gry<br>2. Opis uruchomienia i działania programu<br>3. Szczegóły implementacyjne<br>4. Podział pracy<br>5. Podsumowanie |

1. Opis gry
   Saper jest to klasyczna jednoosobowa gra komputerowa. Jej celem jest odsłonięcie kolejnych pól planszy, a jednocześnie uniknięcie trafienia na minę. Każda komórka planszy zawiera albo liczbę od 0 do 8 (liczba min wokół), albo minę. Gra daje możliwość wyboru poziomu trudności.

2. Opis uruchomienia i działania programu
   Program nie przyjmuje argumentów wywołania. Do uruchomienia gry można skorzystać z komendy make test (do kompilacji – make). Na początku prosi użytkownika o podanie poziomu trudności oraz współrzędnych startowego pola. Następnie, program daje możliwość oflagowania lub odsłonięcia konkretnej komórki. Finalnie, użytkownik informowany jest czy gra została zakończona sukcesem oraz zostaje wyświetlony ranking 5 najlepszych wyników.

3. Szczegóły implementacyjne
   Program wykorzystuje nagłówki conio.h i Windows.h, więc działa jedynie w systemie operacyjnym Windows.
   Najważniejsze struktury:
   - cell_t: struktura przechowująca informacje o konkretnej komórce tj. np. czy jest odsłonięta, czy ma minę, ile jest min wokół.
   - board_t: struktura przechowująca informacje o całej planszy tj. np. jej wymiary, tablicę komórek i obecny stan gry.

   Najważniejsze funkcje:
   - generate_cell: funkcja generująca „pustą” komórkę.
   - generate_board: funkcja tworząca planszę do gry na podstawie podanych wymiarów; używa do tego pomocniczych funkcji np. fill_mines (losuje ustawienie min), generate_connections (łączy sąsiednie komórki), count_mines (zlicza liczbę min w sąsiadujących komórkach).
   - make_move: funkcja wykonująca ruch na planszy; używa do tego pomocniczych funkcji np. flag_cell (oflagowuje dane pole), reveal (odsłania daną komórkę).
   - best_results: zapisuje wynik gry do pliku oraz wypisuje 5 najlepszych wyników.
   - game_from_file: funkcja czyta przykładową gry z pliku i wyświetla jej przebieg wraz z planszą.

4. Podział pracy
   Szymon - silnik gry, wczytywanie przykładowej gry z pliku
   Magda - implementacja rozgrywki, makefile, zapisywanie wyników w pliku

5. Podsumowanie
   Udało się zaimplementować wszystkie zaplanowane funkcjonalności. Rozwiązaliśmy wszystkie napotkane problemy - przykładowo, zastosowaliśmy funkcję fgets zamiast scanf, żeby prawidłowo przyjmować od użytkownika wartości liczbowe, z którymi funkcja scanf miała problemy; zastosowaliśmy też pomocniczą strukturę do przechowywania wyniku gracza razem z jego pseudonimem, żeby możliwe było poprawne sortowanie wyników malejąco. Uważamy jednak, że prawdopodobnie niektóre z funkcjonalności mogły zostać zaimplementowane w bardziej wydajny czasowo lub pamięciowo sposób - ze względu na ograniczenia czasowe, zdecydowaliśmy się na prostsze rozwiązania.