Gra – „Saper”

Zasady gry

W grze „Saper” generowana jest plansza, z losowo umieszczonymi minami. Celem użytkownika jest odkrycie wszystkim pól planszy, które nie są zaminowane. Po odkryciu pola niebędącego miną wyświetla się na nim liczba (od 1 do 8) informująca o liczbie pól z minami sąsiadujących z danym polem (także po przekątnej). Jeżeli po odkryciu pole jest puste oznacza to że nie sąsiaduje z żadnym zaminowanym polem. W przypadku, gdy gracz odkryje pole z miną, gra kończy się niepowodzeniem.

Użytkowanie programu

Program uruchamiamy przy pomocy wybranego terminala (np. PowerShell) przy użyciu polecenia ./Minesweeper będąc w folderze z plikami z programu. Program może być uruchomiony w dwóch trybach:

a) Standardowy – tryb w którym to użytkownik gra w sapera, a program odświeża odpowiednio planszę po wpisaniu poleceń. Aby uruchomić ten tryb należy wpisać ./Minesweeper <ścieżka do pliku>. Następnie gracz proszony jest o wybór poziomu trudności (1 z 3 domyślnych lub utworzenie własnego). Po zakończeniu gry program prosi gracza o podanie swojej nazwy, a następnie dodaje go do listy 5 najlepszych graczy jeżeli jego wynik kwalifikuje się do tego rankingu. Na koniec program wyświetla tę listę.
b) Tryb czytania z pliku – tryb, w którym program wczytuje planszę z pliku podanego przez użytkownika, a następnie wykonuje polecenia podane przez użytkownika w tym pliku. Aby uruchomić ten tryb należy wpisać w terminalu ./Minesweeper -f <ścieżka do pliku>. Program przetwarza wszystkie polecenia odświeżając planszę po każdym. Na koniec, jeżeli polecenia doprowadziły do wygranej lub porażki podaje odpowiednią informację, wynik oraz liczbę poprawnych kroków.

Polecenia, które są akceptowane przez program:

- r <x> <y> - odkrycie pola o współrzędnych x, y. Gdy zostaną podane współrzędne pola już odkrytego program nie wykona żadnej operacji. Jeżeli zostanie odkryte pole z miną gra kończy się porażką.
- f <x> <y> - postawienie flagi na polu o współrzędnych x, y. Flaga służy do oznaczenia pola, w którym użytkownik spodziewa się miny, a tym samym zabezpiecza te pole przed przypadkowym odkryciem. Jeżeli polecenie zostanie wykonane na polu, które ma flagę to flaga zostanie z niego usunięta.

x – nr wiersza, y – nr kolumny

Plansza oraz komendy w pliku w trybie odczytu z pliku powinny mieć następujący format:

1 * * 1 0 1 1 1 0

1 1 1 1 0 1 * 1 0

0 1 1 1 0 1 2 2 1

0 2 * 3 0 0 1 * 1

0 2 * * 2 1 2 2 2

1 2 3 3 * 1 1 * 1

1 * 1 1 1 1 1 1 1

1 1 1 0 0 0 0 0 0

r 1 1

f 1 2

r 1 4

r 1 5

Liczby oznaczają ilość sąsiadujących pól z minami, a * oznaczają pole z miną. Komendy są wypisane bezpośrednio pod planszą.

Szczegóły implementacyjne

Program został podzielony na moduły odpowiadające za różne funkcje. Moduły te to:

main.c

Główny moduł sterujący całym programem. Odpowiada on za zweryfikowanie trybu w jakim użytkownik uruchomił program, a następnie uruchomienie odpowiednych dla danego trybu funkcji.

board.c

Moduł ten zawiera funkcje odpowiedzialne za generowanie planszy, wczytanie jej z pliku czy wizualizację planszy w terminalu. Funkcje zawarte w tym module:

- choose_difficulty – funkcja prosi użytkownika o podanie poziomu trudności, a następnie ustawia odpowiednią liczbę wierszy, kolumn i min na planszy.
- generate_board – funkcja alokuje pamięć na planszę o podanych wymiarach, a także ustawia parametry pól na 0. Funkcja zwraca utworzoną planszę

- generate_mines – funkcja rozmieszcza losowo określoną liczbę min na planszy, a następnie ustawia numery informujące o liczbie min w polach sąsiadujących. Rozmieszczanie min zawarte jest w pęli while, która działa dopóki liczba rozmieszczonych min będzie równa podanej liczbie min. Funkcja dla każdego pola sprawdza ile pól sąsiadujących z nim zawiera minę i przypisuje odpowiedni numer.
- print_board – funkcja odpowiedzialna za wizualizację planszy w terminalu. Poza planszą drukowane są też liczba punktów oraz liczba możliwych do użycia flag. Funkcja zawiera też parametr *isfinished*. Jeżeli jego wartość jest ustawiona na 1 oznacza to, że plansza ma wyświetlić miejsca w których znajdują się miny. Wartość równa 0 oznacza wyświetlanie planszy w trybie gry. Numery w polach są dodatkowo wyświetlane w różnych kolorach (jak w oryginalnym saperze)
- load_board – funkcja, która wczytuje planszę z pliku jeżeli program wywołano z flagą *-f*. W pierwszej pęli while funkcja liczy ilość rzędów i kolumn planszy w pliku, natomiast w drugiej wczytuje liczby lub miny i przyporządkowuje je do odpowiednich pól. Funkcja zwraca utworzoną planszę
- free_board – funkcja zwalniająca pamięć zaalokowaną.

game.c

Moduł ten zawiera funkcje odpowiedzialne za obsługę komend podczas gry w sapera i odpowiednie odświeżanie planszy. Funkcje w tym module:

- are_all_read – funkcja sprawdzająca czy wszystkie pola niebędące minami zostały odczytane. Zwraca wartość 0 lub 1. Zwrócenie wartości równej 1 jest równoznaczne z zakończeniem gry zwycięstwiem.
- read_fields – funkcja odpowiedzialna za odkrywanie pola po użyciu komendy *r*. Dodatkowo jeżeli numer odkrytego pola jest równy 0 to odkrywa ona rekurencyjnie wszystkie sąsiadujące z nim pola niebędące minami. Mechanika ta zawarta jest w oryginalnym saperze i pozwala na usprawnienie odkrywania planszy. Parametr *mode* określa czy odczytywana jest pierwsza komórka (0), czy któraś z kolei(1).
- execute_commands – funkcja działająca tylko w standardowym trybie programu. Miny są generowane po pierwszym wprowadzeniu komendy *r*, aby zapobiec przegraniu w pierwszym kroku. Pobiera ona od użytkownika komendy (*r* lub *f)* i wykonuje związane z nimi operacje. Po każdej operacji funkcja czyści terminal i wyświetla odświeżoną planszę. Funkcja działa w pętli while aż do zakończenia gry. Zwraca 0 – w przypadku porażki – lub 1 – w przypadku zwycięstwa.
- execute_commands_from_file – funkcja działa tylko w trybie czytania z pliku. Operacje wyglądają tak samo jak w komendzie execute_commands z tym, że komendy są odczytywane z podanego pliku a nie wprowadzane przez użytkownika. Program działa do momentu zwycięstwa, porażki lub końca komend w pliku.

results.c

Moduł ten zawiera funkcje związane z obsługą listy najlepszych graczy. Zawarte w nim funkcje to:

- load_result – funkcja wczytuje wyniki najlepszych graczy z pliku (jeśli istnieje), a następnie wpisuje je do wektora struktur. Dodatkowo liczy ilość graczy w podanym pliku i zwraca tę wartość. Jeżeli plik jest pusty lub nie istnieje – zwraca 0.
- add_new_result – funkcja wpisuje nowy wynik gracza do listy 5 najlepszych, jeżeli jego wynik się kwalifikuje. Następnie przy pomoc *qsort* ustawia graczy w wektorze od najlepszego do najgorszego.
- save_result – funkcja nadpisuje podany plik lub tworzy go (jeśli nie istnieje) i zapisuje w nim zaktualizowaną listy.

Program został wyposażony także w kilka struktur.

- Field – struktura pola planszy, zmienne w niej zawarte informują o liczbie sąsiadujących pól z minami, odczytaniu pola, zaminowaniu pola oraz postawionej fladze na polu.
- Difficulty – struktura poziomu trudności. Zawiera liczbę wierszy, kolumn, min oraz wartość mnożnika punktów
- Player – struktura danych gracza. Zawiera nazwę gracza oraz jego liczbę punktów

Podsumowanie

Program spełnia wszystkie założenia projektu. Logika działania gry saper jest zachowana, plansza jest generowana w sposób czytelny. Program działa w dwóch trybach (standardowym saperze i czytaniu z pliku), a także obsługuje plik z listą najlepszych graczy. Największą trudność sprawiła implementacja funkcji czytających planszę oraz komendy z pliku.

Autor projektu: Daniel Kaźmierczak