Sprawozdanie

Zaimplementowanie gry Saper

Wykonała: Nikola Dujka

Spis treści

Podręcznik użytkownika programu

Opis zasad gry

Gra w Sapera polega na odkrywaniu pól na planszy, na której znajdują się ukryte miny. Celem jest odkrycie wszystkich pól, które nie zawierają min, bez ich detonowania. Każde pole, które nie zawiera miny, może ujawniać liczbę sąsiednich min. Gracz musi użyć tej informacji, aby unikać min i wykonać bezpieczne ruchy. Gra kończy się, gdy gracz odkryje wszystkie bezpieczne pola lub jeśli przypadkowo kliknie pole z miną.

Sposób uruchomienia programu

Uruchomienie programu przez **MSYS2 MINGW64**:

Należy przejść do folderu z grą:	**cd ścieżka_do_folderu**
Skompilować pliki za pomocą Makefile:	**make**
Następnie:	
1. wywołać program:	**./saper**
lub	
2. wywołać program z flagą -f:	**./saper -f ścieżka_do_pliku**

Opis poszczególnych opcji

Po wywołania programu za pomocą:

1. ./saper

Użytkownik zostaje poproszony o wybranie poziomu (łatwy, średni, trudny, własna plansza), które różnią się liczbą kolumn, wierszy oraz liczbą min. W przypadku własnej planszy, użytkownik zostaje poproszony o wpisanie własnych liczb kolumn, wierszy i min. Następnie program prosi gracza o odkrycie pierwszego pola (**r liczba liczba**). Gra trwa dopóki użytkownik nie odkryje wszystkich pól lub nie trafi na minę. Po zakończeniu gry gracz zostaje poproszony o imię, po czym zostaje wpisany wraz ze swoim wynikiem do pliku **wyniki.txt**, z którego program następnie pokazuje 5 najlepszych graczy.

2. ./saper -f ścieżka_do_pliku

Program wyświetla planszę wczytaną z pliku, liczbę punktów, poprawnych kroków oraz czy gra zakończyła się sukcesem (1 – wygrana, 0 – przegrana).

Przykładowy schemat pliku:

1 2 1 1 1
1 * 2 2 *
2 3 4 * 2
1 * * 2 1
1 2 2 1 0

r 1 1
r 2 2

Szczegóły implementacji programu

Podział programu na moduły

1. Moduł **sapermain.c**

Moduł ten jest główną częścią programu. Tworzy tablicę W, która jest tablicą wyświetlaną użytkownikowi oraz tablicę T, która zawiera informacje o polach (numery oraz bomby). Za pomocą **getopt** sprawdza, czy została podana flaga -f wraz z plikiem. Jeżeli tak program wykonuje funkcję **funkcjaf()**. Jeżeli nie program pyta użytkownika o poziom trudności i przypisuje następnie odpowiedną liczbę wierszy, kolumn i min.

Następnie program prosi gracza o pierwszy ruch i wchodzi do pętli **while**, która trwa dopóki gra nie zostanie zakończona (zmienna globalna **D** nie będzie wynosić 0).

W pętli wykonywane są funkcje **wyświetl_plansze()** i **wczytaj_polecenie()**. Jeżeli wszystkie pola są odsłonięte lub zmienna D została zmieniona, gra się kończy i program prosi użytkownika o imię. Jest ono wpisywane do pliku **wyniki.txt** wraz z wynikiem. Następnie program wypisuje 5 najlepszych graczy z podanego wyżej pliku.

2. Moduł **plansze.c**

Zawiera 4 funkcje:

- **liczba_pol()**

Iteruje po komórkach tablicy W (wyświetlanej użytkownikowi) i liczy pola, które są odsłonięte i nie są flagą.

- **ile_bomb()**

Liczy dla danej komórki, ile znajduje się wokół niej bomb.

- **stworz_plansze()**

Tworzy planszę z uwzględnieniem, że na podanych wartościach pierwszego ruchu nie znajdzie się bomba. Dopóki aktualna liczba min nie równa się ogólnej liczby min - za pomocą **rand()** losuje kolumnę i wiersz. Jeżeli nie jest to położenie pierwszego ruchu, wstawia tam minę.

W następnej części funkcji sprawdza liczbę min dla każdej komórki za pomocą **ile_bomb()** i wpisuje to do tablicy.

- **wyswietl_plansze()**

Wyświetla planszę zamieniając numer oznaczający flagę (11) na „f”, numer oznaczający bombę (10) na „*”, numer oznaczający pole nieodsłonięte (9) na „#”. Ponadto wyświetla planszę

w 5 kolorach w zależności od rodzaju komórki – biały (#), niebieski (1-8), turkusowy (0), czerwony (*), zielony (f).

3. Moduł **polecenia.c**

Zawiera 3 funkcje:

- **wyk_r()**

Jest funkcją rekurencyjną.

Jeżeli na podanych wartościach komórki znajduje się liczba 1-8 pole zostaje odsłonięte (W[i][j] = T[i][j]).

Jeżeli znajduje się tam liczba 0 wywołujemy **wyk_r()** na komórkach wokół z odpowiednimi założeniami (m. in. tym, że dana komórka musi być nieodsłonięta – inaczej funkcja powtarzałaby się w nieskończoność).

Jeżeli znajduje się tam bomba gra zostaje zakończona (zmiana globalnej zmiennej D) i program wyświetla informację o przegranej oraz pełną planszę.

- **wczytaj_polecenie()**

Wczytuje polecenie z konsoli. Możliwe polecenia to **r** i **f**.

W przypadku polecenia **r** program sprawdza, czy podane liczby są prawidłowe oraz wywołuje dla nich **wyk_r()**.

W przypadku polecenia **f** program sprawdza, czy podane liczby są prawidłowe oraz wstawia w daną komórkę flagę lub ją usuwa.

- **wczytaj pierwsze()**

Wczytuje pierwsze polecenie z konsoli, wymuszając na użytkowniku, by było to polecenie r, a następnie tworzy planszę z pomocą **stworz_plansze()**. Jeżeli użytkownik podał inne polecenie, funkcja wypisuje informację o tym oraz wykonuje się ponownie.

4. Moduł **top.c**

Zawiera strukturę oraz jedną funkcję.

- **topka()**

Wczytuje z pliku wyniki i imiona graczy oraz wpisuje je w odpowiednie miejsca struktury jednocześnie sortując. Następnie wypisuje dane ze struktury.

5. Moduł **funkcjaf.c**

Zawiera dwie funkcje.

- **wyk_r2()**

Jest to zmodyfikowana wersja funkcji wyk_r(). Różnicą pomiędzy dwoma funkcjami są czynności wykonywane w przypadku przegranej. W funkcji_r2() jest to zmiana globalnej zmiennej GRA na 1.

- **funkcjaf()**

Tworzy plansze W – z aktualnym wyglądem planszy i T – ze wszystkimi minami i numerami. Czyta planszę z pliku i wpisuje ją do T. Następnie dopóki plik się nie zakończy wczytuje i wykonuje po kolei polecenia podane w pliku za pomocą **wyk_r2()**. W międzyczasie zlicza liczbę poprawnych kroków i sprawdza, czy gra kończy się wygraną czy przegraną. Na końcu wypisuje liczbę poprawnych kroków, liczbę punktów i wynik gry.

Opis i wynik poszczególnych testów

1. **Test programu z flagą -f i plikiem**

Dla pliku:

1 2 1 1 1
1 * 2 2 *
2 3 4 * 2
1 * * 2 1
1 2 2 1 0

r 1 1
r 2 2

Program wypisuje poprawną planszę (taką jak w pliku). Liczbę poprawnych kroków: 1. Liczbę punktów: 1. Czy wygrana: 0.
Program działa poprawnie.

2. **Test funkcji stworz_plansze() i ile_bomb()**

Dla danych: w=5, k=5, lm=5, x=1, y=1

Przykład wypisywania programu (gdzie 10 to mina):
0 0 0 0 0
0 0 0 0 0
1 2 3 2 1
2 10 10 10 2
2 10 5 10 2

Program działa poprawnie.

3. **Test funkcji topka()**

Dla danych w pliku:

180 eda
72 dfeac

200 qedsa
40 dacz

Program wypisuje:

Top 5 graczy:

1. 200 qedsa
2. 180 eda
3. 72 dfeac
4. 40 dacz
5. 0

Program działa poprawnie.

Podsumowanie

Udało się zaimplementować wszystkie funkcjonalności. Dłużej zastanawiałam się nad tym jak zrobić, by gra cały czas trwała, a przegrana była sprawdzana w funkcji wyk_r(), a nie w main() jak w przypadku wygranej. Rozwiązałam to wprowadzając zmienną globalną D. W wyk_r(), gdy trafimy na bombę, wartość D jest zmieniana, a w main() jest sprawdzana wartość D, dzięki czemu gra zakończy się, nawet jeśli przegrana jest sprawdzana w wyk_r(). Inną trudnością było getopt, ponieważ nie znałam wcześniej tej funkcji. Rozwiązanie – doedukowanie się na ten temat. Poza tymi program nie sprawił mi większych problemów, a sam projekt uważam za ciekawy i inspirujący. Dobrze sprawdził moje umiejętności programowania.