Sprawozdanie Saper
autor: Anna Guz 337561

Podręcznik użytkownika programu

Opis zasad gry

Saper to gra, której celem jest odkryć wszystkie pola na planszy, nie trafiając na minę. Po odkryciu danego pola, numer w tym miejscu oznacza ilość min otaczających to pole. Na podstawie tych informacji można wywnioskować, w których miejscach znajdują się miny, a które należy odkryć.

Sposób uruchomienia programu

Program można uruchomić w dwóch opcjach. W obu przypadkach należy na początku skompilować kod, co można zrobić wpisując do konsoli komendę “make”.
Aby zagrać w grę samodzielnie należy odpalić grę wpisując komendę ./saper.exe bez żadnych dodatkowych argumentów wywołania.
Aby rozpocząć grę w trybie czytania z pliku, należy wpisać do konsoli taką samą komendę oraz dodać flagę -f po czym nazwę pliku. Komenda wygląda następująco:

./saper.exe -f nazwa_pliku

Plik, z którym można odpalić kod powinien być zapisany w poniższym formacie:

n m
[plansza]
komenda1
komenda2
komenda3
…
komendan

gdzie n oraz m to wielkości planszy (n - liczba wierszy, m - liczba kolumn), [plansza] to graficznie przedstawiona plansza, gdzie 9 to bomba, a każda liczba mniejsza od niej to ilość bomb na około danego pola (jeżeli pole nie graniczy z żadną miną, ma ono wartość 0). komenda1, komenda2, … to kolejne komendy dla programu, umożliwiające odsłanianie pól lub stawianie flag na podanych polach.

Komendy

Program akceptuje 2 możliwe komendy:

- r x y - odkrywa pole o numerze wiersza x i numerze kolumny y. Jeżeli na tym polu jest ustawiona flaga, usuwa ją, ale nie odkrywa pola.

- f x y - stawia flagę na polu x y, gdzie x to numer wiersza a y to numer kolumny pola, na którym użytkownik chce postawić flagę. Flaga ułatwia grę przez graficzne odznaczenie tego miejsca jako zajęte. Pomaga także przy wpisywaniu komend. Jeżeli wpiszemy komendę odkrycia pola z flagą, flaga zostanie usunięta, jednak pole nie zostanie odkryte, co ułatwia nie popełnienie błędu.

Szczegóły implementacji programu

Program został podzielony na 6 modułów, wliczając plik główny. W każdym module znajdują się inne funkcje, podzielony funkcjonalnościami. Dostępne moduły:

- saper - główny moduł programu. W jego skład wchodzi jedynie funkcja main, w której wywoływane są wszystkie inne funkcje.
- naj_wyniki - w tym module znajdują się funkcje, które wypisują listę 5 najlepszych graczy i ich wyników oraz dodają aktualny wynik gracza do listy.
- plansza - w jej skład wchodzą funkcje, które wypisują i tworzą plansze a także funkcja odkrywająca pole na planszy.
- poczatek_gry - znajdują się tam funkcje odpalane na początku gry - pobieranie poziomu trudności od użytkownika oraz ustalanie wartości zmiennych dla gry samodzielnej i poprzez plik.
- podsumowanie - posiada funkcje potrzebne do wypisania podsumowania na koniec ruchu - wypisujące aktualny czas od rozpoczęcia gry i aktualny wynik.
- ruch - przechowuje funkcje dotyczące samej gry - pobieranie komend od gracza oraz wykonywanie ich.
- uproszczenie_kodu - moduł z funkcjami, które upraszczają kod i programowanie, sprawiają, że wygląda on schludniej. W jej skład wchodzą funkcja czyszcząca konsolę oraz druga, wypełniająca tablicę zerami.

Interfejs kluczowych funkcji

Najważniejszą funkcją programu jest funkcja pobierz_i_wykonaj_komendy. Funkcja działa w pętli, pobierając komendy dopóki gracz nie przegra (nie odkryje miny) lub odkryje wszystkie pola i wygra. W pętli na początku program pobiera komendę. W zależności od trybu gry pobiera ją albo z konsoli (wpisaną przez gracza) albo z pliku. Cały program działa na podstawie trzech tablic dwuwymiarowych:

- plansza - przechowuje informacje o wytworzonej planszy. Każde pole ma przypisaną cyfrę oznaczającą ilość min dookoła tego pola, lub 9 oznaczające bombę na danym polu
- flagi - tablica zapisująca miejsce flag. Jeżeli na danym polu jest flaga, wartość tego elementu wynosi 1, w innym przypadku jest to 0
- odkryte - tablica z takimi samymi wartościami co flagi. 1 oznacza, że dane pole zostało już odkryte, 0 o tym, że jest jeszcze zakryte i gracz nie wie, co tam będzie.

Korzystając z tych trzech tablic funkcja umożliwia granie w grę. Jeżeli dane pole ma wartość 1 we wszystkich tych tablicach, oznacza to, że jest odkryte, więc nie widać na nim flagi (flagi

pojawiają się jedynie na zakrytych polach). To pole jest wypisywane jako 1 (wartość z tablicy plansza).

Reszta funkcji jest krótsza oraz ma mniej skomplikowane działanie.

Struktury

W programie wykorzystałam trzy struktury. Wszystkie ułatwiają pobieranie danych oraz ich przetwarzanie. Wykorzystywałam je też do łatwiejszego dzielenia kodu na funkcje i moduły.

- lb - struktura odnosząca się do “leaderboard” czyli do listy pięciu najlepszych graczy i ich wyników. Przy pobieraniu tych wyników z pliku najłatwiej mi było zapisać je w strukturze, która posiada wartość string, w której przechowuję nazwę użytkownika oraz int, w której zapisałam wynik danego gracza.
- dane - tę strukturę stworzyłam, aby kod był przejrzysty oraz aby pobieranie komendy od gracza lub z pliku odbywało się w oddzielnej funkcji. Struktura przechowuje wszystkie zmienne, które potrzebowałam w funkcji pobierz_i_wykonaj_komendy, jednak chciałam je pobierać w oddzielnych funkcjach, dlatego struktyra ułatwia przekazywanie tych danych między funkcjami.
- zmienne - wykorzystane do tego samego co struktura dane. Dzięki tej strukturze pobieram zmienne od użytkownika (wielkość tablicy) oraz ilość min na planszy w oddzielnej funkcji, dopiero potem przekazując dane tam, gdzie są potrzebne.

Testy

W pliku Makefile dostępna jest opcja make test, która testuje działanie niektórych funckji. Przez interakcje z użytkownikiem nie było to możliwe we wszystkich funkcjonalnościach dlatego poniżej opiszę dokładniejsze testy całego programu:

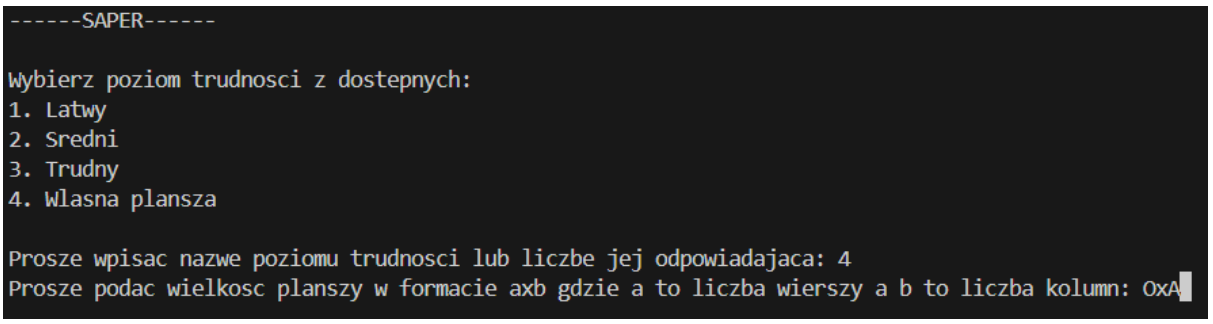
Testy użytkownika wybierania trudności programu:

```
C:\Users\3ania\OneDrive\Pulpit\python\ABCD\c\jimpp\saper\saper.exe: unknown option -- a
Nieznany argument wywolania. Aby otrzymac pomoc wpisz -h.
```

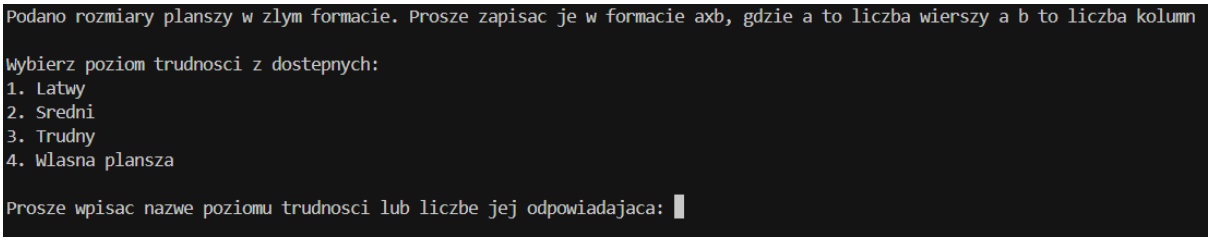
Zdjęcie 1. Wywołanie programu z nieznaną flagą (./saper.exe -a). Program wypisuje wiadomość o tym, że ta flaga jest nieznana oraz daje informacje zwrotną o opcji pomoc.

```
Help: Aby uruchomic gre w trybie czytania z pliku nalezy podac flage -f i podac nazwe pliku.
Aby grac samodzielnie nie nalezy podawac zadnej flagi ani argumentow.
```

Zdjęcie 2. Wywołanie programu z flagą -h. Wyświetla się pomoc uruchamiania programu. Testy flag przebiegły pomyślnie.

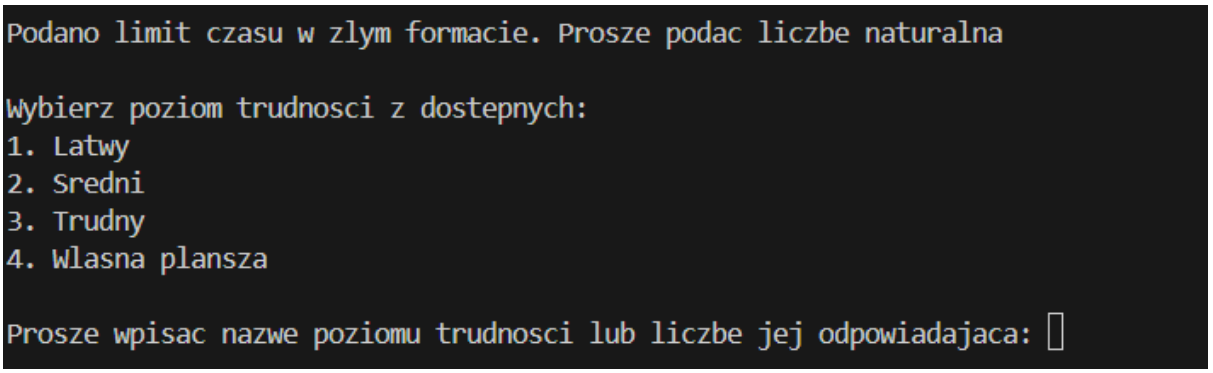


Zdjęcie 3. Próba wywołania programu z opcją trudności “Własna plansza” i nieakceptowanymi danymi wielkości.



Zdjęcie 4. Informacja zwrotna, pokazująca użytkownikowi w jakim formacie należy podawać dane.

W przypadku podania nieprawidłowej wartości czasu (którą też program pobiera od użytkownika otrzymujemy poniższy komunikat:



Zdjęcie 5. Komunikat wypisywany przez program po wpisaniu wartości czasu w złym formacie (jeżeli nie jest liczbą lub jest ujemna).

Podsumowanie

Stworzenie gry od początku wydawało mi się ciekawym pomysłem. Po zaczęciu implementacji zadania nie zmieniło się moje podejście. Zaprogramowanie działającego kodu okazało się przyjemne, jednak dosyć czasochłonne.
Największy problem sprawiło mi dodanie możliwości gry poprzez plik oraz połączenie tego z już działającym kodem reszty programu. Jednak analizując fragmenty kodu po kolei udało mi się wszystko połączyć i zaprogramować w taki sposób, aby działało.
Kolejnym fragmentem, który wydawał mi się trudny było dzielenie programu na moduły. Jako, że do tej pory zazwyczaj pisałam kod w jednym pliku i dzieliłam go jedynie na funkcje

obawiałam się, że rozdzielanie go do różnych plików może się okazać nie wygodne. Tak się jednak nie stało, a dzięki mniejszej ilości linijek w każdym pliku, kod jest bardziej czytelny.