Politechnika Śląska
Wydział Matematyki Stosowanej
Kierunek Informatyka

Gliwice, 14.01.2023

Programowanie I

projekt zaliczeniowy

"***Saper***"

Radosław Rzepka gr. lab. 1

1. Opis projektu

Projekt jest konsolową interpretacją słynnej gry Saper. Celem gry jest odkrywanie na planszy poszczególnych pól w taki sposób, aby nie natrafić na minę. Na każdym z odkrytych pól napisana jest liczba min, które bezpośrednio stykają się z danym polem (od jednej do ośmiu; jeśli min jest zero to na polu nie ma wpisanej liczby). Należy używać tych liczb, aby wydedukować gdzie znajdują się miny. Każde pole można również oflagować, aby oznaczyć sobie gdzie może znajdować się potencjalnie mina. Gra jest wygrana, gdy gracz poprawnie oflaguję każdą znajdującą się na mapie bombę, nie oflagowując żadnego innego pola. Gra jest przegrana, gdy gracz odkryje minę.

2. Wymagania

- menu, sterowane przez użytkownika strzałkami oraz przyciskiem enter, gdzie aktualnie wybrana opcja jest zaznaczona kolorem zielonym
- możliwość włączenia gry na odpowiednim poziomie początkującym, średnim lub zaawansowanym lub możliwość samodzielnego wybrania szerokość, wysokości mapy oraz ilość min, która się na niej znajduje, walidacja tych danych wprowadzonych przez użytkownika
- ranking, pokazujący wyniki gracza wraz z jego nazwą, podział rankingu na poziomy, posortowanie wyników względem czasu potrzebnego do przejścia gry, dane przechowywane są w pliku tekstowym „ranking.txt”
- logika gry
    - dynamiczne generowanie tablicy dwuwymiarowej struktur „Field”
    - losowe generowanie bomb, po pierwszym naciśnięciu przez użytkowania klawisza odkrywającego pole
    - możliwość oflagowywania lub oflagowywania odpowiednich pól
    - odkrywanie poszczególnych pól
    - sprawdzanie czy gracz nie odkrył bomby (przegrana)
    - sprawdzanie czy gracz poprawnie oflagował wszystkie bomby (wygrana)
- obliczanie czasu przechodzenia danej mapy
- możliwość dodania przy wygranej na jednym z predefiniowanych poziomów swojego wyniku do rankingu

3. Przebieg realizacji

Cała aplikacja składa się z 7 plików:

1. **main.cpp** – odpowiedzialny za pierwsze włączeniu menu przy włączeniu programu oraz za poprawne wyświetlanie polskich znaków w programie

2. **menu.cpp** – odpowiedzialny za obsługę menu, poprawne przesuwanie wskaźnika wskazującego na odpowiednio wybraną opcję w menu, oraz za wywołanie odpowiedniej części programu po zatwierdzeniu danej opcji
3. **game.cpp** – odpowiedzialny za całą logikę gry oraz za zapisanie po wygranej grze danych do pliku zawierającego ranking
4. **customSettings.cpp** – odpowiedzialny za włączenie gry z niestandardowymi ustawieniami dotyczącymi szerokość, wysokości mapy oraz ilości min
5. **gameRules.cpp** – odpowiedzialny za wyświetlenie zasad gry i instrukcji sterowania
6. **ranking.cpp** – odpowiedzialny za odczytanie rankingu z pliku tekstowego i odpowiedniego go wyświetlenie na ekranie
7. **functions.cpp** – zawiera funkcję, które używane są w wielu różnych plikach

Przy pisaniu aplikacji skorzystałem z poniższych bibliotek:

- **iostream** – odpowiada za standardowe operacje wyjścia, czyli wypisywanie danych na ekran
- **fstream** – odpowiada za możliwość wczytania danych do pliku i odczytania danych z pliku
- **windows.h** – odpowiada za ustawienie polskich znaków w konsoli, zmiana koloru tekstu w konsoli
- **conio.h** – dostarcza funkcję getch() i kbhit(), która umożliwia nasłuchiwanie jakie znaki na klawiaturze wprowadza
- **algorithm** – dostarcza funkcję sort(), która służy do posortowania wyników w rankingu
- **vector** – umożliwia tworzenie kontenerów typu vectora, do których są wpisywane dane z pliku
- **chrono** – umożliwia mierzenia czasu trwania gry
- **ctime i random** – umożliwiają generowanie liczb pseudolosowych

Program rozpoczyna swoje działania od wywołania funkcji „menu”. W menu można poruszać się strzałkami oraz enterem zatwierdzić wybraną opcję. Opcja podświetlona zielonym kolorem mówi o wybranej aktualnie opcji.

Funkcja „ranking” odpowiada za odpowiednie odczytanie danych z pliku i dodanie ich do odpowiedniego wektora. Później odbywa się ich sortowanie i wypisanie na ekranie.

Funkcja „customSettings” odpowiada za przyjęcie od użytkownika niestandardowych danych dotyczących wielkości planszy i ilości min, walidację tych danych oraz wywołanie funkcji „game” z odpowiednim parametrami

Funkcja „game” odpowiada za całą logikę gry. Na samym początku wypisywana jest plansza, która nie ma jeszcze wylosowanych bomb. Bomby są losowane dopiero przy pierwszym naciśnięciu przycisku „Q” na klawiaturze (odpowiedzialnego za wyświetlanie planszy) na

planszy. Bomby generowane są na całej mapie za wyjątkiem pól w pobliżu naciśniętego przez użytkownika pola (tak jak w oryginalnej grze). Cała plansza jest przechowywana w dynamicznie generowanej tablicy dwuwymiarowej, której elementami są struktury „Struct”

```
struct Field {
    int howManyBombsNear = 0;
    bool hasBomb = false;
    bool isRevaled = false;
    bool isFlagged = false;
};
```

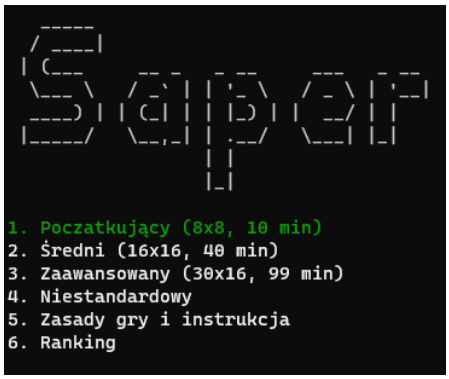
Po wygenerowaniu bomb, odpowiednia funkcja zlicza dla każdego pola na którym nie znajduje się bomba, ile bomb znajduje się wokół danego pola. Pętla while odpowiada za sprawdzanie który przycisk zostanie przez gracza naciśnięty i w ten sposób wykonuje odpowiedniej operacje. Przy naciśnięciu przycisku odpowiedzialnego za wyświetlanie pól funkcja „revealFields” decyduje, które pola mają być odsłonięte. Funkcja wykorzystuję rekurencję aby odsłonić odpowiednią ilość pól.

```
void revealFields (Field **board, int boardWidth, int boardHeight, MoveCords userMove) {
    if(board[userMove.rowCords][userMove.colCords].isRevaled) return;

    board[userMove.rowCords][userMove.colCords].isRevaled = true;

    for (int i = -1; i ≤ 1; i++) {
        for (int j = -1; j ≤ 1; j++) {
            MoveCords newCord;
            newCord.rowCords = userMove.rowCords + i;
            newCord.colCords = userMove.colCords + j;

            if (correctCords(newCord.rowCords, newCord.colCords, boardWidth, boardHeight)) {
                if (board[newCord.rowCords][newCord.colCords].howManyBombsNear == 0 && !board[newCord.rowCords][newCord.colCords].hasBomb && !board[newCord.rowCords][newCord.colCords].
                isRevaled) {
                    revealFields(board, boardWidth, boardHeight, newCord);
                }
                if (!board[newCord.rowCords][newCord.colCords].hasBomb) board[newCord.rowCords][newCord.colCords].isRevaled = true;
            }
        }
    }
}
```

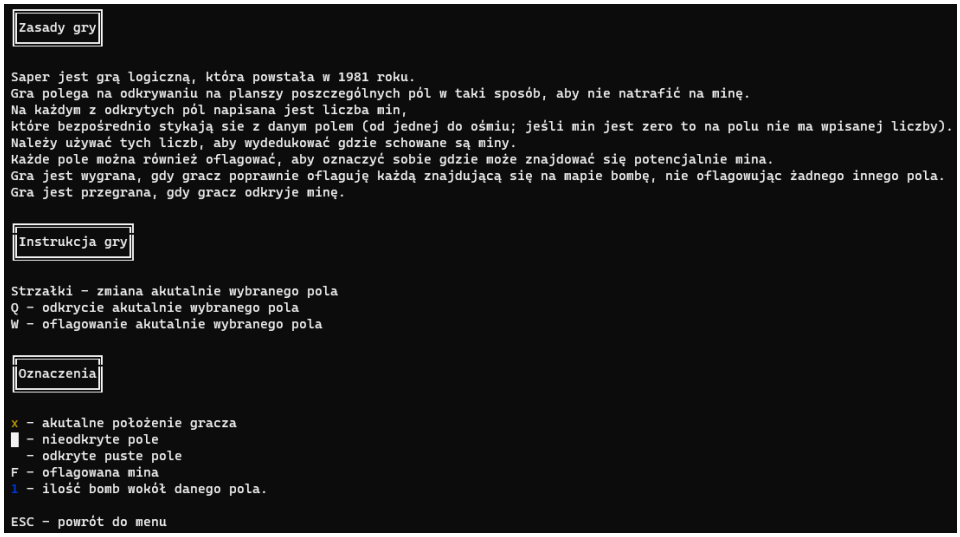
Przy naciśnięciu przycisku odpowiedzialnego za wyświetlanie pól sprawdzane jest również to czy gracz nie odsłonił bomby, co kończy grę, wyświetlając odpowiedni komunikat i umożliwiając powrót do menu.

4. Instrukcja użytkownika

Menu główne

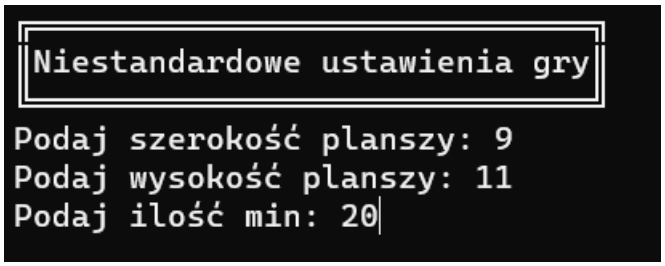


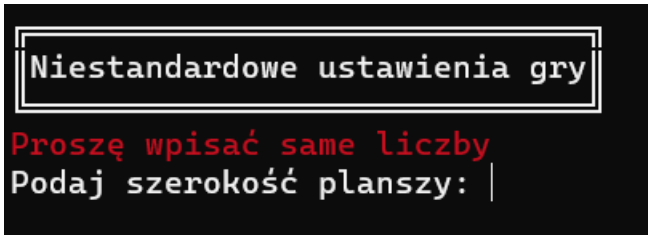
Zasady gry i instrukcja



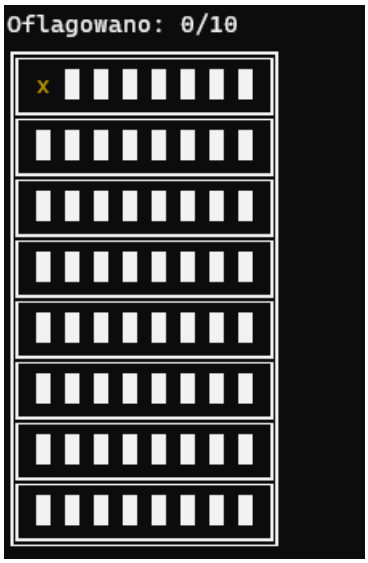
Ranking

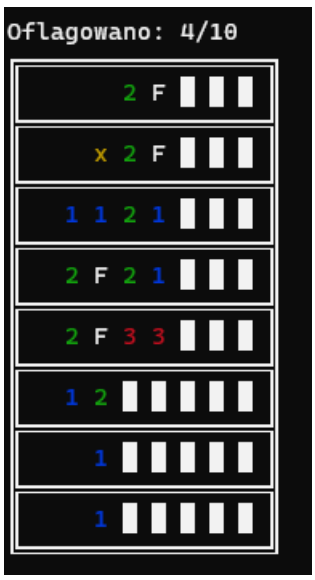
Niestandardowe ustawienia gry

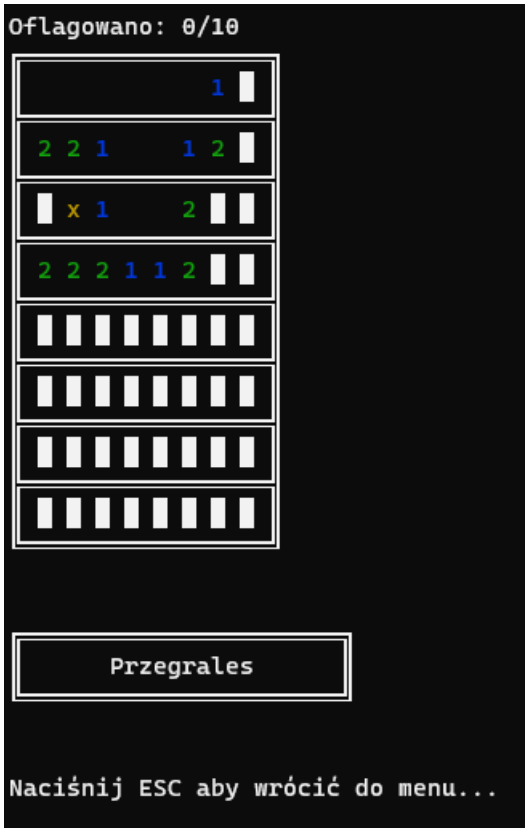




Gra






Oflagowano: 0/10
Przegrales
Naciśnij ESC aby wrócić do menu...


Oflagowano: 10/10
Gratulacje wygrales
Udało ci się wygrać na poziomie początkującym
Twój czas rozwiązywania to: 40 sekund
Podaj swój nick, aby zapisać twój wynik do rankigu (maksymalnie 20 znaków):

5. Podsumowanie i wnioski.

Udało zrealizować mi się wszystkie zaplanowane przed rozpoczęciem projektu plany. Największe problemy miałem z wymyśleniem algorytmu na wyświetlanie odpowiednich pól oraz na obsługę menu. W dalszym etapie można byłoby stworzyć bazę danych, która przechowywałaby dane o wynikach, dzięki czemu każda osoba mająca pliki gry mogłaby wpisywać swoje dane do bazy przez co ranking nie byłby jedynie rankingiem tylko na danym urządzeniu, ale dla każdego gracza.