Dokumentacja projektu - „Gra saper”

- Implementacja sapera:

Saper zaimplementowany jest jako klasa o nazwie `saper`:

```
class saper
{
    vector< vector<int> > stan;
    vector< vector<char> > plansza;

    const int WYM_X;
    const int WYM_Y;

    const int ILOSC_MIN;

    enum { MINA = -1, PUSTE = 0 };

    int wybierz_ilosc_min() const;
    void inicjalizuj_plansze();
    int ile_min_wokol(const int x, const int y) const;
    void odkryj_obszar(const int x, const int y);
    void ustaw_wspolrzedne(int& x, int& y) const;
    void umiesc_flage();
    bool odslon_pole();
    bool czy_wygrana() const;

public:
    saper(int wym_x = 10, int wym_y = 10);
    void rysuj_plansze(bool pokazac_miny = false) const;
    bool ruch_gracza();
};
```

Dane składowe obejmują: zmienne oznaczające wymiary planszy (w programie posługujemy się tylko planszami 10x10), ilość wylosowanych min, nienazwany typ `enum` przechowujący dwie stałe liczbowe oraz dwa wektory wektorów.

Pierwszy z nich, o nazwie `stan`, przechowuje stan planszy w formie liczb. Jego elementy symbolizujące pola mogą przyjmować następujące wartości: -1 – oznaczającą minę na danym polu, 0 – oznaczającą puste pole, oraz wartości od 1-8 oznaczające łączną ilość min na sąsiednich polach.

Zamiast wektorów można było zdefiniować następujące tablice wielowymiarowe:

```
int stan[10][10];
char plansza[10][10];
```

lecz ich wymiary muszą być znane już w czasie kompilacji, tymczasem dzięki wektorom, wymiary mogą być podane w czasie działania programu, np. przez użytkownika. Sposób odwoływania się do elementów tablic dwuwymiarowych przedstawionych powyżej oraz do elementów dwóch wektorów jest w naszym przypadku identyczny.

Opis funkcji składowych klasy:

- **`int saper::wybierz_ilosc_min() const`**

  Funkcja wywoływana w konstruktorze, prosząca gracza o wybranie ilości min do umieszczenia na planszy, z zakresu od 1 do 20.

- **`void saper::inicjalizuj_plansze()`**

  Funkcja wywoływana w konstruktorze, odpowiedzialna za inicjalizację nowej planszy, tj. losowe rozmieszczenie odpowiedniej ilości min oraz uzupełnienie dwóch wektorów odpowiednimi wartościami.

- **`int saper::ile_min_wokol(const int x, const int y) const`**

  Funkcja zliczająca ilość sąsiadujących min z danym polem o współrzędnych `[x, y]`. Sprawdza ona występowanie min na polach o współrzędnych:

  ```
  [x-1, y-1][x, y-1][x+1, y-1]
  [x-1, y  ][x, y  ][x+1, y  ]
  [x-1, y+1][x, y+1][x+1, y+1]
  ```

- **`void saper::odkryj_obszar(const int x, const int y)`**

  Funkcja stosująca rekurencyjne wywołania aby odsłonić hurtowo sąsiadujące pola, które nie zawierają min ani flag. Rekurencyjne wywołania tej funkcji korzystają z takich samych argumentów jak współrzędne przedstawione w powyższej funkcji `saper::ile_min_wokol`.

- **`void saper::ustaw_wspolrzedne(int& x, int& y) const`**

  Funkcja pobierająca od gracza dwie współrzędne i sprawdzająca je pod obecność błędnego formatu.

- **`void saper::umiesc_flage()`**

  Funkcja umieszczająca flagę na polu wskazanym przez gracza.

- **`bool saper::odslon_pole()`**

  Funkcja odsłaniająca pole wskazane przez gracza i zwracająca odpowiednią wartość, stosowną do zawartości pola. Jeśli gracz natrafił na minę, funkcja zwraca wartość `false`. W innym przypadku zwraca wartość `true` oznaczającą, że gra toczy się dalej.

- **`bool saper::czy_wygrana() const`**

  Funkcja sprawdzająca czy plansza jest tak uzupełniona, że pozwala nam na zwycięstwo i zakończenie aktualnej gry.

- **`saper::saper(int wym_x, int wym_y)`**

  Konstruktor odpowiedzialny jest za inicjalizację stałych składowych klasy, odpowiednie zwiększenie wektorów do podanych wymiarów planszy, oraz wywoływanie funkcji `saper::inicjalizuj_plansze`.

- **`void saper::rysuj_plansze(bool pokazac_miny) const`**

  Funkcja rysująca planszę na ekranie, jeśli przekazany argument ma wartość `true`, funkcja zaznacza na planszy miejsce występowania wszystkich min (jest to przydatne przy kończeniu gry).

- **`bool saper::ruch_gracza()`**

  Funkcja odpowiedzialna za interakcję z graczem, zwraca wartość `true` jeśli gra toczy się dalej lub `false` jeśli aktualna gra się zakończyła (gracz wygrał lub przegrał).

- Przebieg gry:

Cała gra ma miejsce w obrębie funkcji `main`. Aby rozpocząć rundę, wystarczy zdefiniować obiekt typu `saper` i wywoływać na rzecz tego obiektu funkcję `saper::ruch_gracza` w pętli, aż do zwrócenia przez nią wartości `false` (świadczącej o wygranej lub przegranej gracza):

```
saper moja_gra;

while(moja_gra.ruch_gracza());
```