

1. Autor i data wykonania oraz nazwa gry

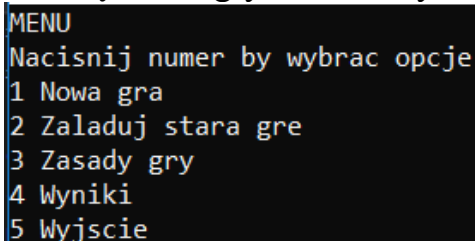
Autorem gry o nazwie Hungry Penguin jest Aneta Kopera, powstała ona 24.05.2019r.

2. Temat gry

Gra opiera się na prostej planszy dwuwymiarowej, na której znajdują się dwa typy pól. Typami tymi są kra lodowa (*) oraz dziura (-). Użytkownik steruje postacią pingwina wyświetlaną w wersji konsolowej jako P, musi kierować nim za pomocą klawiatury i chodzić tylko po krach lodowych. Jego zadaniem jest nakarmienie pingwina czyli doprowadzenie go do posiłku – ryby wyświetlanej jako R. W grze jest ważny czas w jakim użytkownik przejdzie planszę. W przypadku gdy pingwin wejdzie na pole z dziurą jego pozycja jest resetowana do pozycji początkowej, ale czas przejścia nie jest wtedy zerowany, a sumuje się – czyli jest sumą całej próby pokonania planszy. Gdy użytkownik znajduje się na skraj planszy i wcisnie przycisk kierujący go poza planszę to pozycja pingwina pozostanie ta sama.

3. Opis gry

Po włączeniu gry oczom użytkownika ukazuje się ekran menu.



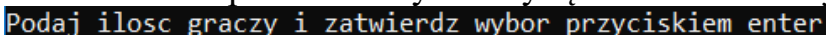
```
MENU
Nacisnij numer by wybrac opcje
1 Nowa gra
2 Zaladuj stara gre
3 Zasady gry
4 Wyniki
5 Wyjscie
```

Rysunek 1: Widok menu

W menu znajduje się 5 opcji. Pierwsza opcja pozwala na zagranie użytkownika w nową grę na losowo wygenerowanej planszy.

- Nowa gra

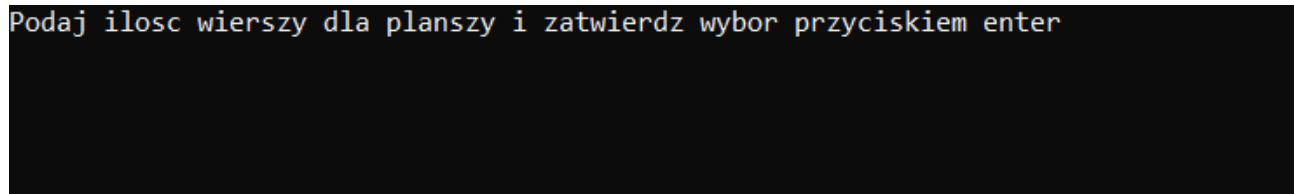
Wybierając opcję nowa gra czyli wciskając 1 następuje przekierowanie do nowej gry. Najpierw wyświetlany jest komunikat o wyborze ilości graczy. Jeżeli wystąpi pomyłka ekran zostanie ponownie wywołany łącznie z informacją o błędzie,



```
Podaj ilosc graczy i zatwierdz wybor przyciskiem enter
```

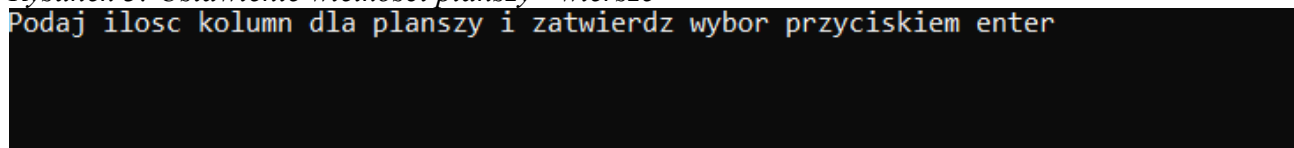
Rysunek 2: Widok wyboru ilości graczy

Następnie wyświetlane są ekrany wyboru wielkości planszy, czyli ilości wierszy i kolumn. Plansze powinny mieć stosunkowo małą liczbę wierszy, a dużą kolumn. (np. 10 - 60). W przypadku pomyłki zostanie ponownie wyświetlony ekran wraz z komunikatem błędu.



```
Podaj ilosc wierszy dla planszy i zatwierdz wybor przyciskiem enter
```

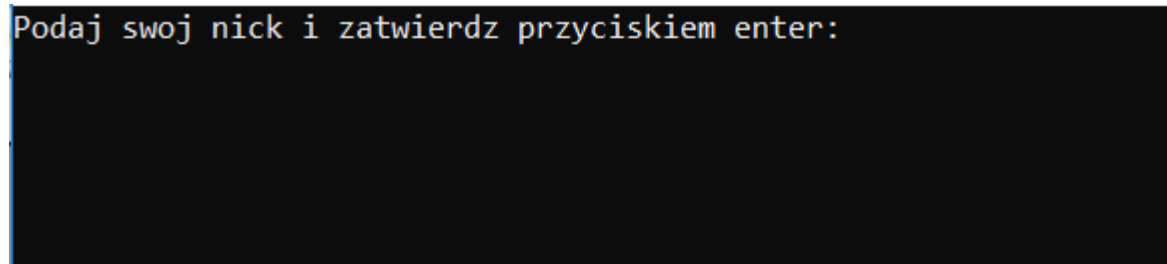
Rysunek 3: Ustawienie wielkości planszy - wiersze



```
Podaj ilosc kolumn dla planszy i zatwierdz wybor przyciskiem enter
```

Rysunek 4: Ustawienie wielkości planszy - kolumny

Kolejno wyświetla się ekran w, którym użytkownik jest proszony o podanie nicku który będzie przypisany do jego wyniku.

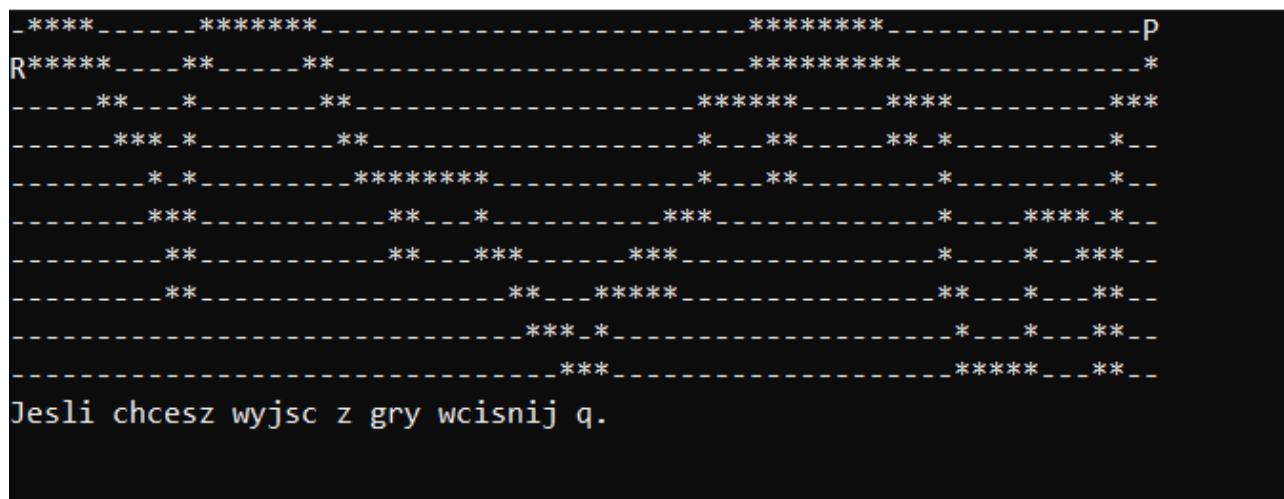


```
Podaj swój nick i zatwierdz przyciskiem enter:
```

Rysunek 5: Ustawienie nicku użytkownika

Ostatecznie po ustawieniach pojawia się ekran odliczania 5 sekund, po upływie tego czasu startuje gra. Po jej zakończeniu – dotarciu do celu, użytkownik zostaje przekierowany do menu głównego, gdzie może sprawdzić swój wynik.

Jeśli jednak została wprowadzona większa liczba graczy niż jeden to po grze jednego użytkownika, następuje przekierowanie do wprowadzenia nicku drugiego użytkownika i znów odliczanie, a następnie ekran gry. Takie czynności trwają dopóki nie zagra ostatni uczestnik. W przypadku gdy któryś z uczestników przerwie grę wynik zapisuje się tylko dla graczy, którzy ukończyli daną planszę.

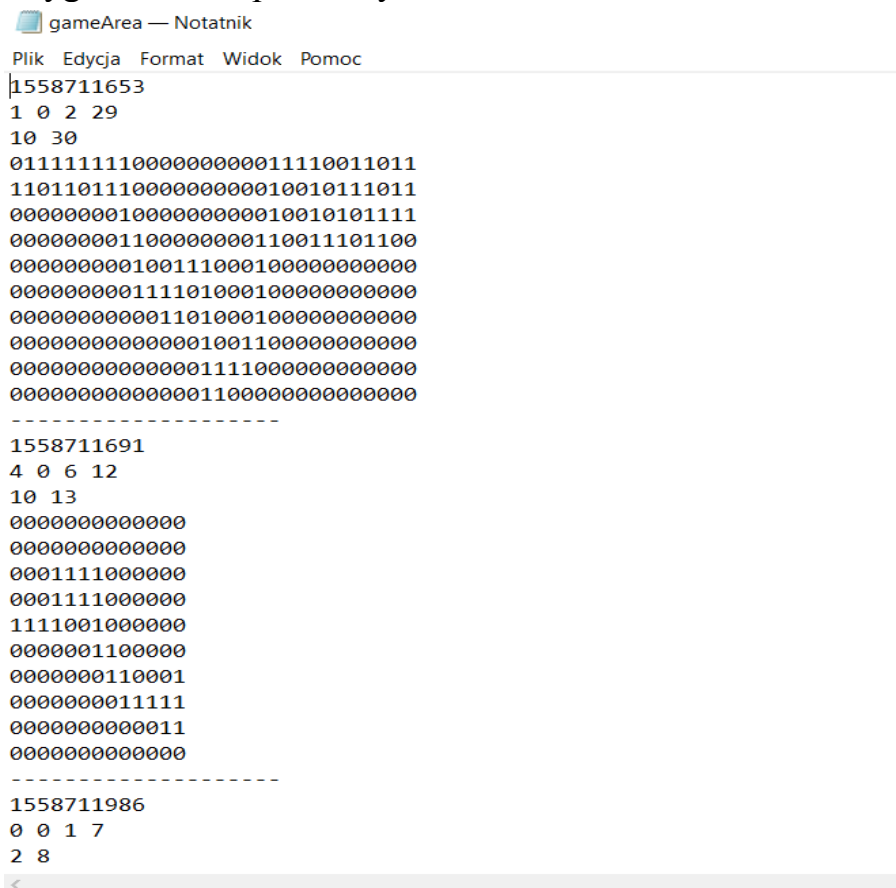


Rysunek 6: Ekran gry

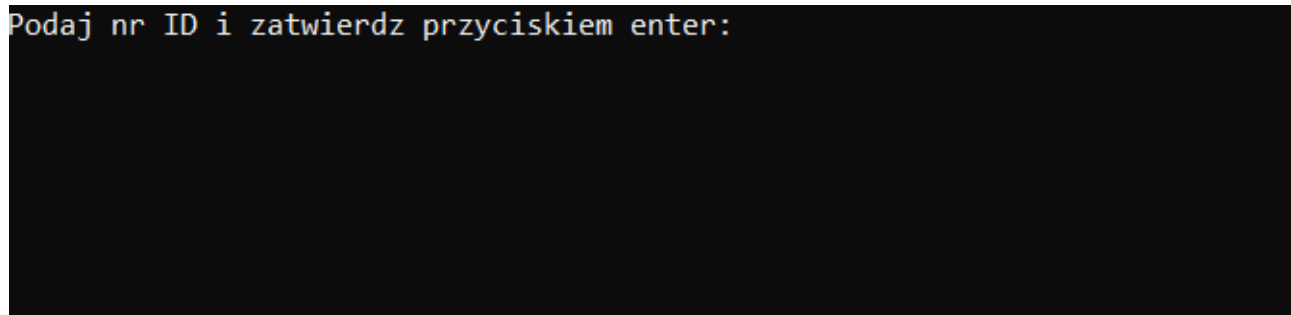
Drugą opcją menu głównego jest załaduj starą grę.

- Załaduj stara gra

Każda plansza podczas generowania się w nowej grze ma nadawane unikalne id i jest eksportowana do pliku gameArea.txt. Dzięki temu użytkownik ma możliwość zagrania planszą, którą kiedyś już grał. Wystarczy, że wpisze jego id, a następnie może wybrać ilość graczy i nick. Plik przechowuje wszystkie plansze kiedykolwiek wygenerowane przez użytkownika.



Rysunek 7: Plik z planszami gry

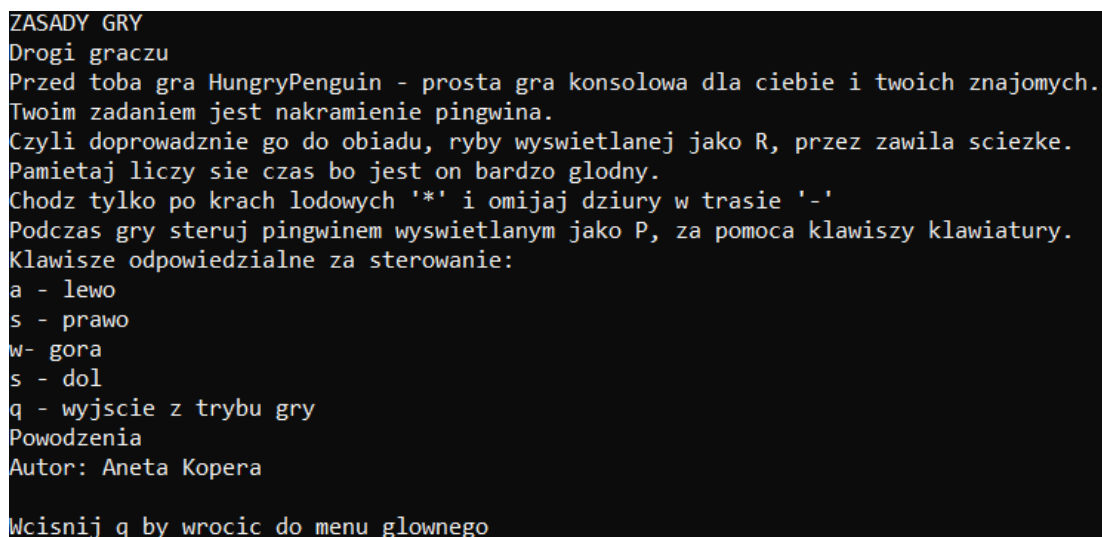


Podaj nr ID i zatwierdz przyciskiem enter:

Rysunek 8: Ekran ładowania gry po id

Trzecim ekranem jest ekran zasad gry. Wyjaśnia on na czym polega gra i jej sterowanie.

- Zasady gry



ZASADY GRY
Drogi graczu
Przed toba gra HungryPenguin - prosta gra konsolowa dla ciebie i twoich znajomych.
Twoim zadaniem jest nakramienie pingwina.
Czyli doprowadzenie go do obiadu, ryby wyswietlanej jako R, przez zawila sciezke.
Pamietaj liczy sie czas bo jest on bardzo glodny.
Chodz tylko po krach lodowych '*' i omijaj dziury w trasie '-'
Podczas gry steruj pingwinem wyswietlanym jako P, za pomoca klawiszy klawiatury.
Klawisze odpowiedzialne za sterowanie:
a - lewo
s - prawo
w- gora
s - dol
q - wyjscie z trybu gry
Powodzenia
Autor: Aneta Kopera

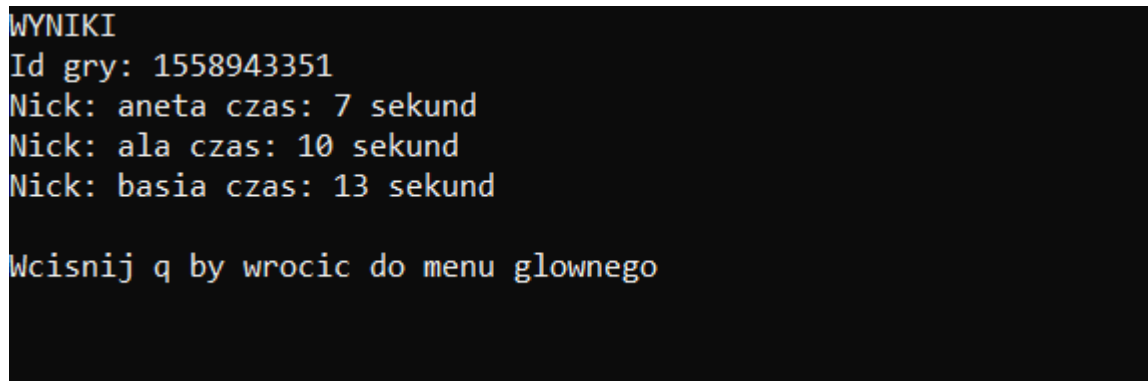
Wcisnij q by wrocic do menu glownego

Rysunek 9: Ekran zasad gry

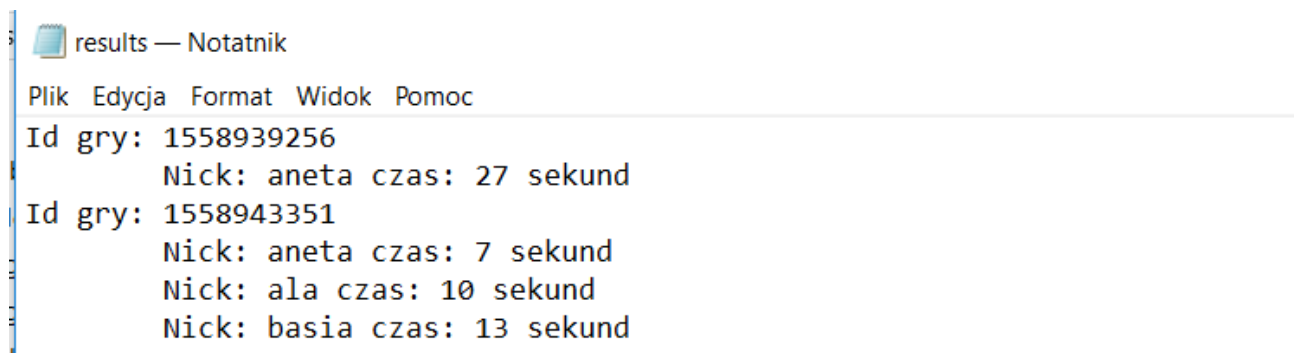
Wyniki gry są przechowywane w opcji 4 menu głównego.

- Wyniki

Wynik znajdujące się w nim dotyczą jednej gry. Natomiast jeżeli użytkownik chce zobaczyć stare wyniki, to ma taką możliwość, ponieważ wyniki po każdej grze są eksportowane do pliku o nazwie resluts.txt znajduje się tam również id gry – opisującej planszę na jakiej został osiągnięty dany wynik.



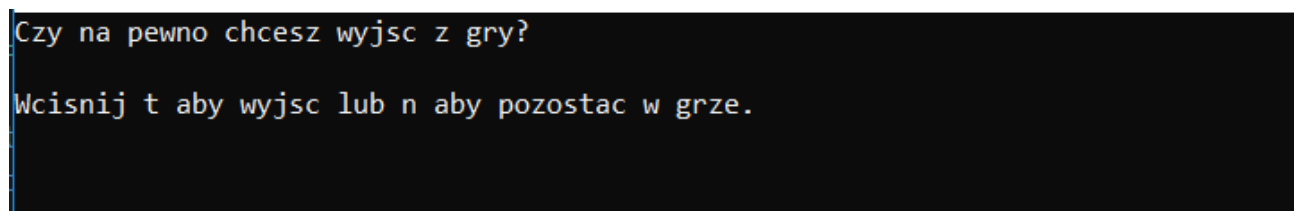
Rysunek 10: Ekran wyników



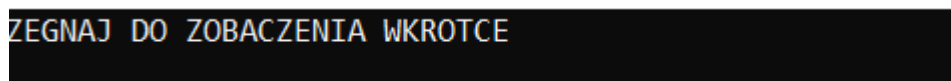
Rysunek 11: Plik zawierający wyniki wszystkich gier

- Wyjście

Ostatnią opcją menu jest wyjście z gry. Po jej włączeniu uruchamiany jest komunikat potwierdzający czy na pewno użytkownik chce wyjść z gry, jeśli wciśnie t, to wyświetlany jest ekran pożegnalny i gra wyłącza się. W przeciwnym wypadku – wciśnięcie n powoduje powrót do menu głównego.

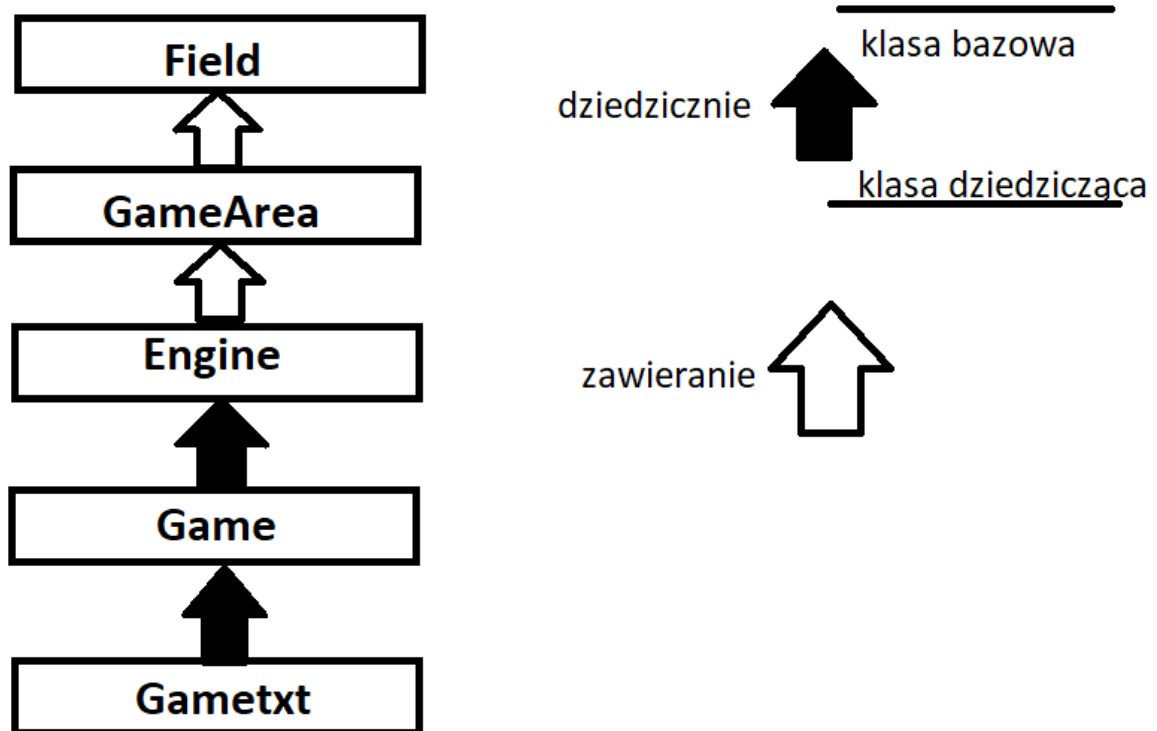


Rysunek 12: Ekran wyjścia z gry



Rysunek 13: Ekran pożegnalny

4.Struktura klas



Rysunek 14: Struktura klas

Klasa Field

protected:

bool status – zmienna boolowska przechowująca informację o komórce (krą lodowa czy dziura)

public:

Field() -konstruktor bezparametrowy

Field(const Field& f) -konstruktor kopiujący

Field& operator=(const Field& f) – operator =

virtual ~Field() - wirtualny destruktor

void setIceFloe() - bezparametrową metodę typu void ustawiającą pole typu Field na krą lodową (1)

bool getInformation() - getter typu bool zwracający informację czy dana komórka jest krą lodową czy dziurą

Klasa GameArea

protected:

*Field **area* – wskaźnik na tablicę dwuwymiarową typu Field (na planszę gry)

int rows – zmienna przechowująca ilość wierszy

int columns – zmienna przechowująca ilość kolumn

public:

GameArea() - konstruktor bezparametrowy ustawiający zmienne

GameArea(int rows, int columns) – konstruktor parametrowy ustawiający zmienne

GameArea(const GameArea& g) -konstruktor kopiujący

GameArea& operator=(const GameArea& g) – operator =

virtual ~GameArea() - wirtualny destruktory niszczący elementy dynamiczne

int getRows() const – getter typu int zwracający ilość wierszy planszy

int getColumns()const – getter typu int zwracający ilość kolumn

void setRows() - setter ustawiający ilość wierszy

void setColumns() - setter ustawiający ilość kolumn

void setField(int X, int Y) – setter ustawiający wskazane pole planszy na krę lodową

bool getField(int X, int Y) – getter zwracający bool – informację o stanie wskazanego pola planszy

Klasa Engineprotected:

*GameArea * gA* – wskaźnik na obiekt klasy *GameArea*

int penguinX- zmienna typu int przechowuje aktualną x pozycję pingwina na planszy

int penguinY- zmienna typu int przechowuje aktualną y pozycję pingwina na planszy

int penguinXstart- zmienna typu int przechowuje początkową x pozycję pingwina na planszy

int penguinYstart - zmienna typu int przechowuje początkową y pozycję pingwina na planszy

int fishX- zmienna typu int przechowująca aktualną x pozycję ryby na planszy

int fishY- zmienna typu int przechowująca aktualną y pozycję pingwina na planszy

public:

Engine() - konstruktor bezparametrowy

Engine(const Engine& e) -konstruktor kopiujący

Engine& operator=(const Engine& e) – operator =

virtual ~Engine() - wirtualny destruktory usuwający dynamiczne zmienne

void setPenguinPosition(int x, int y) -setter ustawiający pozycję pingwina w parametrach podajemy pozycje x oraz y

void setPenguinPositionstart(int x, int y) - setter ustawiający początkową pozycję pingwina od x i y

void setFishPosition(int x, int y) – setter usawiający pozycję ryby na planszy od x i y

int getPenguinPositionX() - getter zwraca aktualną pozycję x pingwina

int getPenguinPositionY() - getter zwraca aktualną pozycję y pingwina

int getFishPositionX() - getter zwraca pozycję x ryby

int getFishPositionY() - getter zwracający pozycję y ryby

int getPenguinPositionXstart() - getter zwracający startową pozycję x pingwina

int getPenguinPositionYstart()- getter zwracający startową pozycję y pingwina

int Analyze(int direction, int x, int y) – metoda analizująca czy dany ruch użytkownika pingwinem jest prawidłowy

Klasa Game

protected:

std::vector<std::pair<std::string, double>> Users – wektor przechowujący nicki użytkowników i ich wyniki czas w sekundach

unsigned long id - zmienna przechowująca id planszy

int usersAmount – zmienna przechowująca ilość użytkowników aktualnie grających

std::fstream gameAreaFile – zmienna fstream do pliku zawierającego wygenerowane plansze

std::fstream gameResultsFile – zmienna fstream do pliku zawierającego historię wyników użytkowników na danej planszy

public:

Game() - konstruktor bezparametrowy

Game(const Game& g) -konstruktor kopiujący

Game& operator=(const Game& g) – operator =

virtual ~Game() - wirtualny destruktor zamykający pliki

virtual void start() - metoda wywołująca rozpoczęcie gry

virtual void newGame() - metoda opcji nowej gry wykorzystuje metody generującą plansze gry, ustawiające parametry i eksportującą planszę do pliku oraz wywołuje zapoczątkowanie gry

virtual void loadOldGame() - metoda pozwalająca na zagranie planszą już kiedyś uzyskano

virtual void generateArea() -generuje losową planszę o parametrach ustawionych przez użytkownika

virtual void play(int usersAmount) – metoda wywołująca gre od ilości użytkowników

virtual void exportArea() - metoda eksportująca do pliku id gry i jej planszę

virtual void exportResults() - metoda eksportująca do pliku wyniki użytkowników

virtual void displayGameRules()=0 – abstrakcyjna metoda wyświetlania zasad gry

virtual void displayMenu() = 0 - abstrakcyjna metoda wyświetlająca menu

virtual char userDecision() = 0 - abstrakcyjna metoda pobierająca znak wcisnięty przez użytkownika na klawiaturze

virtual void displayResults() = 0 - abstrakcyjna metoda wyświetlająca wyniki poprzednio zagranej gry

virtual void displayGame() = 0 –abstrakcyjna metoda wyświetlająca pole gry

virtual void displayExit() = 0 -abstrakcyjna metoda wyświetlająca opcje wyjścia

virtual int move() = 0 abstrakcyjna metoda poruszania się pingwinem zwraca inta, w którą stronę chce się nim użytkownik poruszyć

virtual int countOfUsers() = 0 - abstrakcyjna metoda wyświetlająca ekran podania

ilości użytkowników

virtual int rowSize() = 0 - abstrakcyjna metoda ustawiania rozmiaru wierszy planszy

virtual int columnSize() = 0 - abstrakcyjna metoda ustawiania rozmiaru kolumn dla planszy

virtual void counting() = 0 – metoda odliczająca 5 sekund i wyświetlająca licznik

virtual std::string readNick() = 0 - abstrakcyjna metoda ustawiania niku dla użytkownika

virtual void errorHandling(std::string error) = 0 - abstrakcyjna metoda, która przechwytuje treść błędów wyrzuconych przez metody (z try catch)

virtual void saveUserTime(double userTime, std::string nick,

std::vector<std::pair<std::string, double>> &Users) – metoda zapisująca wyniki użytkowników do pliku

virtual void resetUsers() - metoda usuwająca historię wyników z pamięci programu

virtual unsigned long readID() = 0 - abstrakcyjna metoda czytająca id gry podane przez użytkownika

virtual void loadArea(unsigned long &loadID, unsigned long &actualID, std::string &ignore, int &r, int &c) – metoda ładująca grę na podstawie id (opcja 2 menu)

private:

virtual void parser(int index, std::string numbers) – prywatna metoda przetwarzająca stringa z pliku (ciągu 0 i 1) oznaczających stan pola planszy

Klasa Gametxt

public:

Gametxt() - konstruktor bezparametrowy

Gametxt(const Gametxt& g) -konstruktor kopiujący

Gametxt& operator=(const Gametxt& g) – operator =

virtual ~Gametxt() - wirtualny destruktor

virtual void displayMenu() -metoda wyświetla menu

virtual char userDecision() - metoda wprowadzania inputu użytkownika

virtual void displayGame() - metoda wyświetlająca grę

virtual int move() - metoda do poruszania się klawiszami w grze

virtual void displayGameRules() - metoda wyświetlająca zasady gry

virtual void displayResults() - metoda wyświetlająca wyniki poprzednio zagranej gry

virtual void displayExit() - metoda wyświetlająca ekran wyjścia

virtual int countOfUsers() - metoda wprowadzająca ilość użytkowników

virtual int rowSize() - metoda nadająca pobierająca ilość wierszy od użytkownika dla planszy gry

virtual int columnSize() - metoda nadająca pobierająca od użytkownika ilość kolumn dla gry

virtual void counting() - metoda wyświetlająca odliczanie 5 sekund

virtual std::string readNick() - metoda prosząca użytkownika

o wpisanie swojego niku dla gry

virtual void errorHandling(std::string error) – metoda wyświetlająca przechwycone wyjątki

virtual unsigned long readID() - metoda prosząca użytkownika o wpisanie id gry do załadowania