**Київський національний університет**

**імені тараса шевченка**

**Проєкт на тему 32**

Морський бій

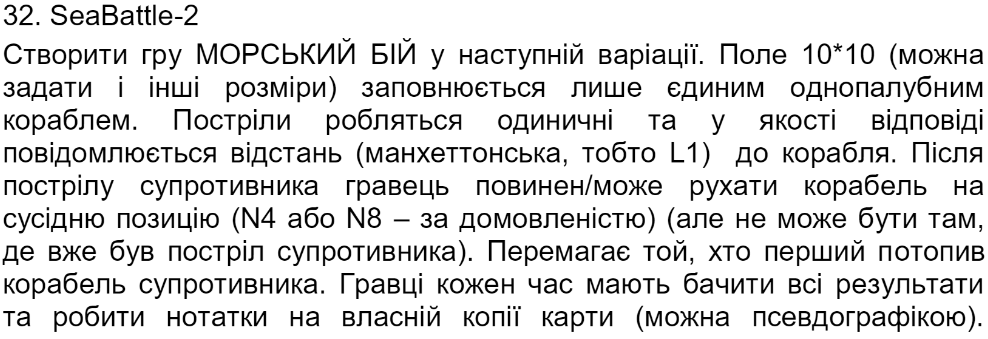
Виконала:

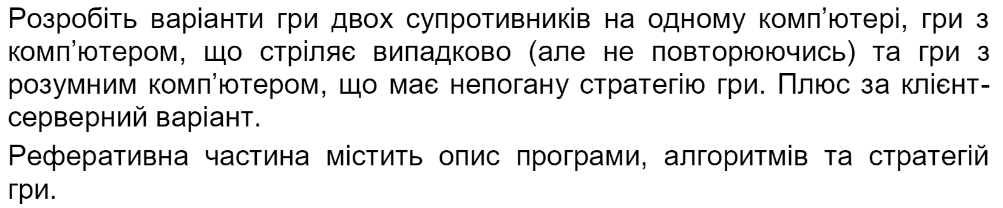
студентка групи Статистика 1

Чернобровкіна Д. О.

Київ 2021

Завдання:





Реалізація:

Класи для гри:

* GameField – об’єкти цього класу - це поля для гри, які мають свою розмірність (NxN), заповнені числами (0 – пуста клітинка, 1 – клітинка, у яку вже був проведений постріл, 2 - кораблик).
* Game – його об’єкти це злиті воєдино 2 об’єкти класу GameField як 2 гравці, та допоміжні методи для зручного маніпулювання грою

Було створено 3 файли:

1. Заголовочний файл SeaBattle32.h з описом вищеперерахованих класів
2. Файл модуля SeaBattle32.cpp, де описані усі функції там методи класів
3. Файл з тестами SeaBattle32\_test.cpp з тестами до описаних класів

Вміст файлів:

1. SeaBattle32.h

Коментарі дають достатній початковий опис.

class GameField

{

private:

    int dimension;

    int \*\*field;

    int playerPosX, playerPosY;

    int maxDistanceToMove;

public:

    GameField();

    // initialization with no parameters

    GameField(int dim, int x, int y, int maxToMove);

    // Initialization with parameters

    // @dim - dimension of field NxN

    // @x - player vertical position

    // @y - player horizontal position

    // @maxToMove - maximum distance that player can move

    ~GameField();

    // Destructor

    int getDimension();

    // @result - field's dimension

    void setDimension(int dim);

    // sets field's dimension

    // @dim - new dimension

    void fieldSetInitial();

    // Creates field and sets on it player's position

    bool canSetPos(int x, int y);

    // Checks if point on field is available

    // @x - vertical position on field

    // @y - horizontal position on field

    // @result - 1 if available, 0 if not

    int setShotPos(int x, int y);

    // Sets point on field as Shot

    // @x - vertical position on field

    // @y - horizontal position on field

    // @result - if point was available, sets point on field

    //           and returs 1, else field stays the same and returns 0

    int setPlayerPos(int x, int y);

    // Sets point on field as Player

    // @x - vertical position on field

    // @y - horizontal position on field

    // @result - if point was available, sets point on field

    //           and returs 1, else field stays the same and returns 0

    void setRandomPlayerPos();

    // Sets any available point on field as Player

    void setRandomShotPos();

    // Sets any available point on field as Shot

    int countMoveDistance(int x, int y);

    // Counts distance to player

    // @x - vertical position on field

    // @y - horizontal position on field

    // @result - int distance from point (x, y) to Player

    bool isPlayerDead();

    // Checks if player is dead

    // @result - 1 if yes, else 0

    void showField(bool hideShip);

    // Shows field on a screen

    // @hideShip - 1 if hide ship, else 0

    // friend std::ostream &operator<<(std::ostream &os, const GameField &field);

    friend std::istream &operator>>(std::istream &is, GameField &field);

    // User enters player position from console

};

1. SeaBattle32.cpp

Основні методи та їх алгоритми:

Клас **GameField**

Заповнення поля нулями там двійкою у тому місці, де знаходиться корабель

void GameField::fieldSetInitial()

{

    this->field = new int \*[this->dimension];

    for (int i = 0; i < this->dimension; i++)

    {

        this->field[i] = new int[this->dimension];

        for (int j = 0; j < this->dimension; j++)

        {

            this->field[i][j] = 0;

        }

    }

    this->field[this->playerPosX][this->playerPosY] = 2;

}

Встановлення нової позиції гравця на полі може відбутись лише коли нова позиція вільна (немає у цій точці пострілу) там дистанція до цієї точки не перевищую дозволену

int GameField::setPlayerPos(int x, int y)

{

    int distance = this->countMoveDistance(x, y);

    if (this->canSetPos(x, y) && distance <= this->maxDistanceToMove)

    {

        this->field[this->playerPosX][this->playerPosY] = 0;

        this->playerPosX = x;

        this->playerPosY = y;

        this->field[x][y] = 2;

        return 1;

    }

    else

    {

        return 0;

    }

}

Метод виводить у консоль поле розміром dimension. У пустих клітинках буде зірочка (\*), у клітинках, де були постріли – хрестик (Х), а корабел зображується кружочком (0)

void GameField::showField(bool hideShip)

{

    std::cout << "     ";

    for (size\_t i = 0; i < this->dimension; i++)

    {

        std::cout << i << " | ";

    }

    std::cout << std::endl

              << std::endl;

    for (int i = 0; i < this->dimension; i++)

    {

        for (int j = 0; j < this->dimension; j++)

        {

            if (j == 0)

                std::cout << i << " |  ";

            std::string toDraw;

            int pos = this->field[i][j];

            if (pos == 2 && !hideShip)

                toDraw = "O";

            else if (pos == 1)

                toDraw = "X";

            else

                toDraw = "\*";

            std::cout << toDraw << " | ";

        }

        std::cout << std::endl;

    }

}

Дружня функція введення об’єкту з консолі забезпечує коректне введення координат гравця. Розмірність та максимальна відстань для ходу має бути задана до цього.

std::istream &operator>>(std::istream &is, GameField &field)

{

    bool cantSetPos;

    field.fieldSetInitial();

    do

    {

        std::cout << "Enter position - ";

        is >> field.playerPosX >> field.playerPosY;

        cantSetPos = !field.canSetPos(field.playerPosX, field.playerPosY);

        if (cantSetPos)

            std::cout << "Enter right coordinates..." << std::endl;

    } while (cantSetPos);

    field.fieldSetInitial();

    return is;

}

Клас **Game**:

Метод для ініціалізації гравців – 2 гравців з консолі, якщо режим гри гравець проти гравця, у іншому випадку одного гравця та одного боту (позиція гравця задається навмання). Гравці є екземплярами **GameField**, тому «вводяться» за допомогою дружньої функції введення з консолі.

void Game::initializePlayerParameters()

{

    if (this->gameMode == 0)

    {

        for (size\_t i = 1; i < 3; i++)

        {

            GameField \*player = this->playerWhoShoot(i);

            std::cout << "Player " << i << std::endl;

            std::cin >> \*player;

        }

    }

    else

    {

        int randX = rand() % this->getDimension();

        int randY = rand() % this->getDimension();

        std::cin >> \*this->playerOne;

        this->playerTwo = new GameField(this->getDimension(), randX, randY, this->maxDistance);

    }

}

Метод show показує двох гравців, але для одного з них він приховує корабель, щоб той, хто стріляє не знав про позицію ворожого корабля.

В залежності від параметра hiddenPlayer він приховує одного чи іншого гравця за допомогою методів класу **GameField::showField**, який також приймає параметр, що дозволяю приховати чи показати гравця.

void Game::show(bool hiddenPlayer)

{

    if (hiddenPlayer)

    {

        std::cout << "  Your field:" << std::endl;

    }

    else

        std::cout << "  Enemie's field:" << std::endl;

    this->playerOne->showField(!hiddenPlayer);

    std::cout << std::endl;

    if (hiddenPlayer)

    {

        std::cout << "  Enemie's field:" << std::endl;

    }

    else

        std::cout << "  Your field:" << std::endl;

    this->playerTwo->showField(hiddenPlayer);

    std::cout << std::endl;

}

Метод **start** починає гру, розмірності та інші параметри про гравців мають бути визначені до виклику методу. Кожного кроку лічільник **counter** збільшується на одиницю. За допомогою нього визначається поточний гравець, що стріляє, та інший, який приймає постріли на своє поле за допомогою методів **PlayerWhoShoot** та **PlayerWhoReceive**,які залежать від лічильника. Далі обраний гравець робить постріл, після якого повідомляється відстань до ворожого корабля. У випадку, коли режим гри «гравець проти гравця», так продовжується поки не потоплять один з кораблів. Якщо ж режим гри проти комп’ютера, то після кожного ходу гравця комп’ютер випадково обирає куди стріляти та куди посунути корабель (стріляє не повторюючись) за допомогою методів **GameField::setRandomShotPos** та **setRandomPlayerPos.**

void Game::start()

{

    int counter = 1;

    std::cout << std::endl;

    while (!this->isGameOver())

    {

        system("CLS");

        this->show(counter % 2);

        GameField \*playerWhoShoot = this->playerWhoShoot(counter);

        GameField \*playerWhoReceive = this->playerWhoReceive(counter);

        std::cout << "  Step " << floor(counter / 2) << std::endl;

        int distance = this->setShotPos(playerWhoReceive);

        std::cout << "Distance was - " << distance << std::endl;

        this->setPlayerPos(playerWhoShoot);

        if (this->gameMode != 0)

        {

// To set next step again to PLAYER | in sum counter += 2, so player will shoot again

            counter += 1;

            if (this->isGameOver()) break;

            // Bot does his operations

            playerWhoShoot->setRandomShotPos();

            playerWhoReceive->setRandomPlayerPos();

        }

        counter += 1;

        std::cout << std::endl;

    }

    if (this->gameMode == 0)

        std::cout << "  Game over, player " << (counter % 2) + 1 << " won!!" << std::endl;

    else

    {

        if (counter % 2 == 0)

            std::cout << "  Game over, player won!!" << std::endl;

        else

            std::cout << "  Game over, bot won!!" << std::endl;

    }

}

Статичний метод завантаження гри з файлу (вміст файлу під кодом).

Game \*Game::loadGame(std::string filename)

{

    int dimension, gamemode, maxDist, x, y;

    std::ifstream file(filename);

    file >> dimension >> gamemode >> maxDist;

    std::cout << dimension << gamemode << maxDist;

    Game \*game = new Game(dimension, gamemode, maxDist);

    file >> x >> y;

    std::cout << x << y;

    game->playerOne = new GameField(dimension, x, y, maxDist);

    file >> x >> y;

    std::cout << x << y;

    game->playerTwo = new GameField(dimension, x, y, maxDist);

    int counter = 1;

    while (!file.eof())

    {

        file >> x >> y;

        game->playerWhoReceive(counter)->setShotPos(x, y);

        file >> x >> y;

        game->playerWhoShoot(counter)->setPlayerPos(x, y);

        counter += 1;

    }

    std::cout << x << y;

    file.close();

    return game;

}

Спочатку з файлу читається розмірність, режим гри, та максимальна відстань для ходу. Далі створюється екземпляр класу **Game game.** Далі манупуляції відбуваються з ним та його двома гравцями. Зчитуються початкові координати гравців, далі один за одним виконують постріли та переміщення кораблів. Після закінчення зчитування повертається **game.** Далі можна почати гру вже з останнього ходу, який відбувся після зчитування файлу.

