**МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ**

**НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «ЛЬВІВСЬКА ПОЛІТЕХНІКА»**

**ІКНІ**

Кафедра **ПЗ**



**ЗВІТ**

до лабораторної роботи №1

**на тему:** *“Розробка алгоритму руху роботів”*

**з дисципліни** *“Конструювання програмного забезпечення”*

**Лектор:**

доц. каф. ПЗ

Сердюк П. В.

**Виконав:**

ст. гр. ПЗ-33

Юшкевич. А.І.

**Прийняв:**

ст. викл. каф. ПЗ

Ковалевич Т. С.

«\_\_» \_\_\_\_\_\_\_\_\_ 2024р.

∑=\_\_\_\_\_

Львів – 2024

**Тема:** Розробка алгоритму руху роботів.

**Мета:** Ознайомлення з засобами розробки Visual Studio та Resharper. Після виконання лабораторної студент повинен уміти створювати проекти, підключати бібліотеки, відлагоджувати програми.

# Теоретичні відомості

**Опис.** На карті розміром 100 на 100 є ресурси енергії та роботи які їх збирають. Кожен студент реалізує алгоритм за яким його роботи рухатимуться, збиратимуть ресурси та створюватимуть нових роботів. Лабораторна є конкурсною, тобто на одній карті будуть одночасно роботи усіх студентів, що здають лабораторну. Кількість балів за лабораторну залежить від кількості ресурсів, який зібрали роботи за 50 ходів усіх роботів.

**Опис конкурсу.**

***Карта*** є квадратною розміром 100x100, тобто її клітинки мають координати (x,y), де x та y можуть набувати значень від 0 до 99. За межі карти вийти неможливо, і при такій спробі робот просто пропускає хід. На одній клітинці може перебувати лише один робот.

***Робота алгоритму.*** Випадковим чином визначається чиї роботи будуть іти першими, чиї другими, третіми і т.д. Спочатку діє перший робот першого учасника, потім перший робот другого учасника, потім перший робот третього учасника, і т.д. Потім діє другий робот першого учасника, другий робот другого учасника, другий робот третього учасника, і т.д. Хід закінчується коли усі роботи виконали дію. Робот може виконати тільки одну з трьох дій: рухатись на якусь клітинку, збирати ресурси на станції, створити нового робота. Для дії робота йому передається уся інформація карти. Деталі реалізації алгоритму нижче.

***Рух роботів*** потребує енергії, яка рівна квадрату відстані між початковою і кінцевою клітинками, та визначається за формулою E = (x1 – x0)2 + (y1 – y0)2 , де (x0, y0) – точка на якій знаходиться робот перед початком руху, (x1, y1) – точка у яку має намір прийти робот***.*** Таким чином, чим швидше рухається робот тим більше енергії він витрачає. Для прикладу: якщо робот переміщується на 5 клітинок по вісі X за один хід, то він витратить 52 = 25 одиниць енергії. Якщо він рухатиметься по лише одній клітинці 5 ходів, то він витратить лише 12 +12+12+12+12=5 одиниць енергії. Якщо у робота менше енергії ніж рух необхідно, то дія робота ігнорується і він залишається на місці. **Увага!** Якщо на клітинці уже стоїть інший (свій або чужий) робот, то робот “спихає” його на сусідню вільну клітинку, витрачаючи при цьому 10 одиниць енергії (якщо у нього вона є, в протилежному випадку він втрачає хід). Робот, якого “спихнули” з клітинки, не витрачає енергії. Можливі певні відмінності, у залежності від варіанту.

***Енергія*** можуть бути отримана із зарядних станцій, які її накопичують та генерують. На клітинці карти може бути розташована тільки одна станція. Для того щоби отримати цю енергію робот повинен стати на клітинку зі станцією і збирати заряд ( або на певній відстані– залежить від варіанту). Максимальна енергія, яку накопичують станції рівна 1000. Кожен хід станції генерують від 10 до 20 одиниць енергії. За один хід робот може зібрати максимально 200 одиниць енергії з одної станції. Для різних варіантів можуть біти відмінності у константах.

На початку станції не мають енергії.

***Створення нових роботів*** також потребує енергії. Новий робот матиме енергію 100 (по замовчуванню, але можна задати інше значення), і з’являється на сусідній вільній клітинці. Робот, що створив нового робота втрачає 200 одиниць енергії + енергію, що є у нового робота (100 по замовчуванню). Для того щоб створити нового робота, у батьківського робота повинно бути не менше цієї енергії, інакше дія робота буде ігноруватись. Рекомендується залишати батьківському принаймі 100 одиниць енергії. Для різних варіантів можуть біти відмінності у константах. Максимальна кількість своїх роботів – 100.

***Початкові умови.*** Кожному учаснику надається 10 роботів із енергією 100, розташованих у випадкових місцях на карті. Кількість станцій є у 5 разів більшою ніж кількість роботів. Станції у початковий відлік часу не мають енергії і тільки починають її накопичувати.

**Варіант 5)** Станцій у 5 разів більше ніж усіх роботів на початку змагання, кожна з яких генерує від 20 до 40 одиниць енергії за хід. При нападі на іншого робота, забирається 5% його енергії і втрачається 50 одиниць енергії. Роботи можуть збирати енергію зі всіх станцій на відстані 1-ї клітинки. Енергія втрат при створенні нового робота рівна 100. За кожні 6 000 очок ставиться 1 бал.

# Принцип розробленого алгоритму

## **Загальний опис принципу:**

На початку гри ми збираємо інформацію про всіх роботів, відділяємо своїх від чужих, зберігаємо інформацію про стан гри.

У роботів є 3 варіанти дій:

* **Рухатися до станції;**
* **Збирати енергію;**
* **Розмножуватися;**

Ці дії містяться в окремих класах: OccupyStationCommand, CollectCommand, BreedCommand, що наслідуються від абстрактного класу Command. Дії асоціюються з роботом за допомогою патерну Strategy. Зміна дії відбувається при виконанні певних умов за допомогою патерну State.

## **Перший етап:**

На першому етапі роботу присвоюється команда OccupyStationCommand. Вона обраховує під капотом відстань до найближчої станції, яка не зайнята своїми і додається до Dictionary де асоціюється з цим роботом. Також обраховується оптимальний шлях до неї. В функції руху врахована можливість руху по поверхні як по Тору. Вираховується оптимальна кількість кроків.

## **Другий етап:**

На кожному кроці перевіряється, чи ми дійшли до станції.

* Якщо ні, знову присвоюємо роботу команду OccupyStationCommand.
* Якщо дійшли – передати нову команду CollectComand

Команда CollectComand збирає енергію. Якщо робота виб’ють зі станції команда віддає керування OccupyStationCommand, щоб робот знову повернувся на свою станцію. Якщо робот має більше ніж 200 енергії і союзниї роботів менше 100 CollectComand передає керування BreedCommand.

## **Третій Етап:**

Команда BreedCommand знову перевіряє умови (Енергія >= 200, Кількість роботів < 100)

* Якщо умови справджуються команда створює нового робота;
* Якщо ні – перевіряє чи ми на станції:
* Так – передає керування CollectCommand
* Ні – передає керування OccupyStationCommand

# Опис Unit-тестів

## **BreedTest**

1. *RobotCollectsThanBreeds:* Перевіряє послідовність дій: дійти до станції - зібрати енергію – розмножитися.

## **CommandSwitch**

1. *BreedToCollect:* Перевіряє, чи коректно відбувається перехід від розмноження до збирання енергії.
2. *BreedToOccupy:* Перевіряє, чи коректно відбувається перехід від розмноження до руху.
3. *CollectToBreed:* Перевіряє, чи коректно відбувається перехід від збирання енергії до розмноження.
4. *CollectToOccupy:* Перевіряє, чи коректно відбувається перехід від збирання енергії до руху.
5. *OccupyToCollect:* Перевіряє, чи коректно відбувається перехід від руху до збирання енергії.

## **MovementTest**

1. *OneRobotGetToStation:* Перевіряє, чи успішно робот дійде до станції.
2. *OneRobotGetToStationLongDistance:* Перевіряє, чи дістанеться робот до станції, до якої він не може дістатися за один крок.
3. *RobotGoToOtherSide:* Перевіряє, чи робот може дістатися до іншого краю карти, рухаючись по площині як по тору.
4. *TwoRobotsSameClosestStation:* Перевіряє, чи не почнуть союзні роботи “боротися” за станцію, яка є для них найближчою і яка знаходиться на однаковій відстані від них.

**Код юніт тестів лежить на GitHub репозиторії разом із кодом.**

# Код програми

Додаю [посилання](https://github.com/Akmitliviy/Robot) на GitHub репозиторій.

**Висновки:** протягом виконання цієї лабораторної роботи я розробив власний алгоритм, використовуючи принципи SOLID та патерни проектування.