**Програмиране на C#.**

**Демонстрация на BFS(Breath First Search) алгоритъм.**

**Автор: Даниел Дамянов**

**Клас: 11А**

**No: 7**

Днес ще ви демонстрирам популярния BFS алгоритъм за търсене в ширина с използването на опашка.

**Задача:** Даден е лабиринт, в който можем да изберем начална точка, от която да тръгнем и крайна точка, до която да стигнем, както и стени, които нямаме право да преминаваме. Да се намери най – краткия път до крайната точка, ако такъв съществува.

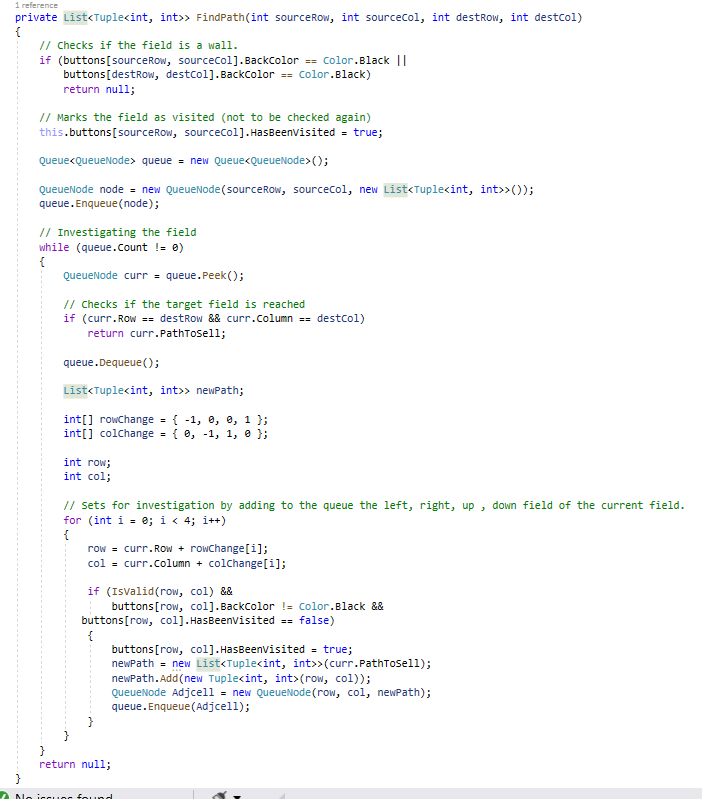
**Анализ на задачата:** Това е класическа backtracking задача, която се решава с търсене в ширина. Използването на рекурсивен метод няма да свърши работа, защото DFS(Depth First Search) търси първо в дълбочина и така губим най-краткия път.

**Инструкции за тест на програмата:**

1. **При отваряне, изберете стартово поле чрез кликане върху лабиринта(Оцветява се в зелено)**
2. **След това, изберете крайно поле чрез кликане върху лабиринта(Оцветява се в червено)**
3. **Изберете стените чрез кликане върху лабиринта(Оцветяват се в черно).**
4. **При кликане върху вече избрана стена тя се изтрива**
5. **Когато сте готови, натиснете FindPath button-a**
6. **Резултата излиза в MessageBox, а най-краткия път е нарисуван с жълт цвят.**
7. **Натиснете ОК и повторете стъпките.**

**Ето примерно решение на задачата:**

Прилагам снимка само от метода, който използва BFS:



Най-ключовото в алгоритъма е добавянето на изследваните клетки в опашка. Така търсенето разглежда полетата в реда на отдалечаването им от стартовата клетка и най-краткия път е гарантиран, което не би станало при рекурсивния DFS.

По пълна документация на всички методи и помощни класове може да намерите source code – a в Github.

Github Repository: <https://github.com/danieldamianov/BFS>

Direct download: <https://github.com/danieldamianov/BFS/archive/master.zip>

Exe:

**Обобщение:** Предложената задача е добър пример за нуждата от използване на BFS с опашка. Така е гарантирано, че ще получим най – краткия път.

**Препоръки:** Задачата може да се разшири чрез имплементирането на други алгоритми за търсене на най – кратък път като ***A\* Search***  или **Dijkstra's** Algorithm. Задачата е направена за не много дълго време, така че има вероятност до се открият бъгове и неточности. Ако откриете такива или имате някакви въпроси от всякакъв тип по задачата ми пишете във Facebook – Daniel Damianov или на email: danieldamianov02@gmail.com