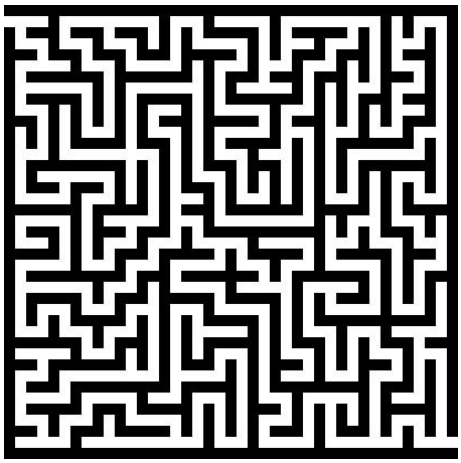
Projektowanie obiektowe

LABORATORIUM 4

Dzisiaj opracowywać będziemy algorytm znajdujący najkrótszą drogę przez labirynt binarny, reprezentowany przez utworzoną przez nas klasę SolveMaze.

W celu wczytania labiryntu z pliku, należy utworzyć prywatną metodę loadMaze, która odczytuje plik tekstowy i tworzy tablicę liczb całkowitych, gdzie −1 reprezentuje ścianę labiryntu, 0 reprezentuje korytarz labiryntu, 1 jest punktem początkowym, −2 jest końcem. Ściany w pliku tekstowym są oznaczone jako *W*, korytarze jako *C,* początek to *S,* koniec to *F*. Znaki są rozdzielane przez tabulator.



Aby znaleźć rozwiązanie, skonstruuj prosty (aczkolwiek mało wydajny) algorytm, badając wszystkie możliwe ścieżki, a następnie cofając się wzdłuż najkrótszej ścieżki. Będziesz mógł poruszać się tylko w czterech kierunkach, w lewo, w prawo, w górę, w dół (ale nie w kierunkach ukośnych). Wykonuj swoje operacje za pomocą tablicy wczytanej przez loadMaze. Zaczynasz od węzła 1, który jest początkiem, a węzły, osiągnięte po wszystkich możliwych pojedynczych ruchach, oznaczasz jako 2. Następnie zaczynasz z każdym węzłem oznaczonym jako 2, a następnie zaznaczasz węzły, osiągnięte po wszystkich możliwych pojedynczych ruchach jako 3 (musisz wykluczyć ściany i węzły, które zostały już odwiedzone). W tym celu należy utworzyć prywatną metodę numberAdjacent, która dla węzła *n* numeruje potencjalne sąsiednie węzły jako *n* + 1. Utwórz publiczną metodę makePaths, która eksploruje labirynt przy użyciu elementu numberAdjacent, aż do osiągnięcia ostatniego punktu. Następnie możesz cofnąć się labirynt, przechodząc od końca do początku, przy każdym ruchu przechodząc do mniejszego numeru węzła. Utwórz publiczną metodę backtrack, która wykonuje wycofywanie i tworzy tablicę zawierającą najkrótszą ścieżkę. Utwórz niezbędne metody wyświetlania wyników (labirynt i najkrótsza ścieżka).