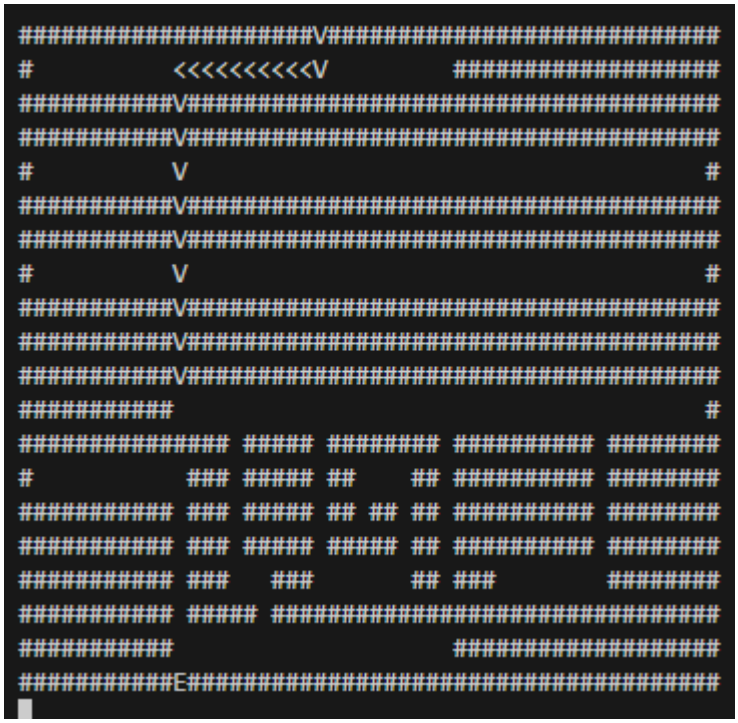
SELENAY HUR

23080103374

Dr Öğretim Görevlisi SELMAN HIZDAL

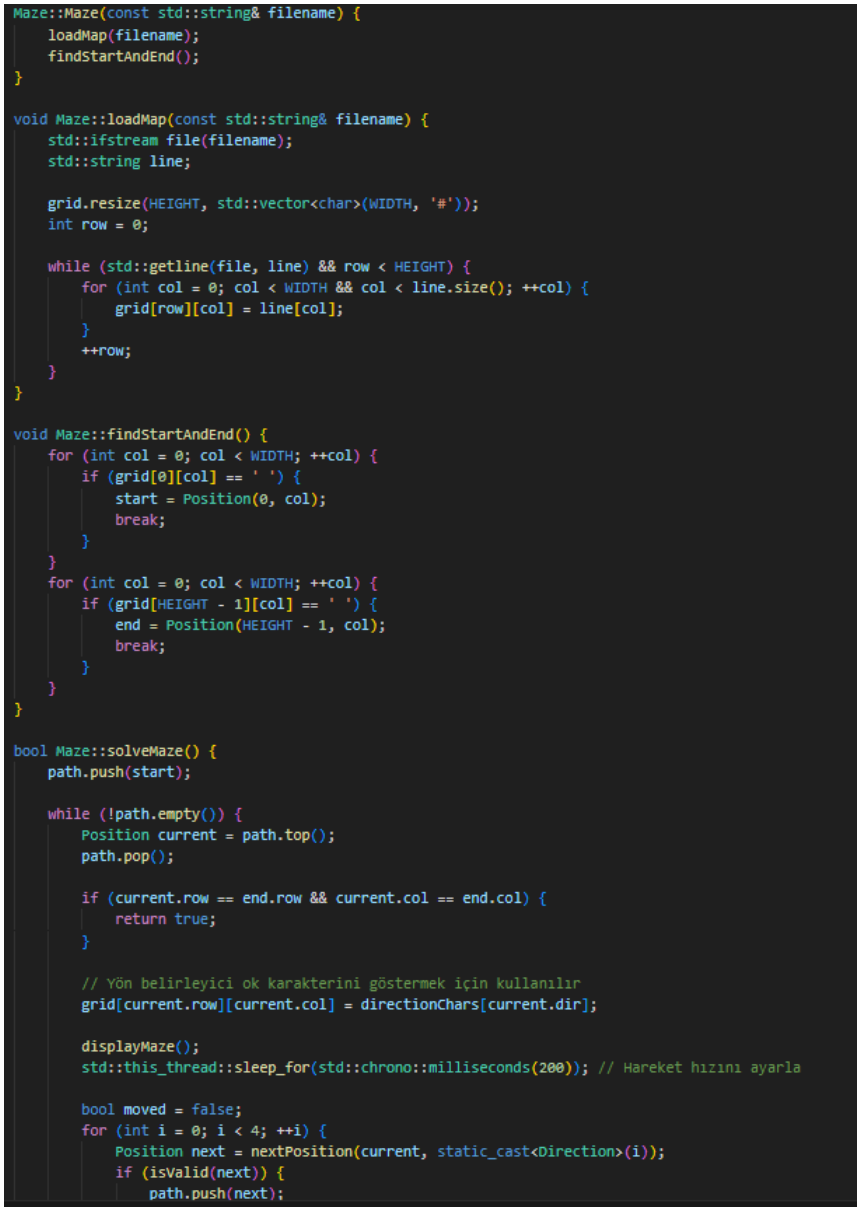


Kodum burada gözüktüğü gibi başlangıç noktasından çıkıyor ve en sonki boşluğa kadar gidiyor .Çıkışta E yazıyor ama harita.txt ye E koymama gerek yok.

```cpp
#include <iostream>
#include "Maze.h"

int main() {
    Maze maze("harita.txt");
    maze.showSolution();
    return 0;
}
```

main(): Bu işlev, Maze sınıfını kullanarak "harita.txt" dosyasından labirenti yükler ve çözümü gösterir.



```cpp
Maze::Maze(const std::string& filename) {
    loadMap(filename);
    findStartAndEnd();
}

void Maze::loadMap(const std::string& filename) {
    std::ifstream file(filename);
    std::string line;

    grid.resize(HEIGHT, std::vector<char>(WIDTH, '#'));
    int row = 0;

    while (std::getline(file, line) && row < HEIGHT) {
        for (int col = 0; col < WIDTH && col < line.size(); ++col) {
            grid[row][col] = line[col];
        }
        ++row;
    }
}

void Maze::findStartAndEnd() {
    for (int col = 0; col < WIDTH; ++col) {
        if (grid[0][col] == ' ') {
            start = Position(0, col);
            break;
        }
    }
    for (int col = 0; col < WIDTH; ++col) {
        if (grid[HEIGHT - 1][col] == ' ') {
            end = Position(HEIGHT - 1, col);
            break;
        }
    }
}

bool Maze::solveMaze() {
    path.push(start);

    while (!path.empty()) {
        Position current = path.top();
        path.pop();

        if (current.row == end.row && current.col == end.col) {
            return true;
        }

        // Yön belirleyici ok karakterini göstermek için kullanılır
        grid[current.row][current.col] = directionChars[current.dir];

        displayMaze();
        std::this_thread::sleep_for(std::chrono::milliseconds(200)); // Hareket hızını ayarla

        bool moved = false;
        for (int i = 0; i < 4; ++i) {
            Position next = nextPosition(current, static_cast<Direction>(i));
            if (isValid(next)) {
                path.push(next);
```

- **Maze(const std::string& filename)**: Bu yapıcı işlev, belirtilen dosyadan labirent haritasını yükler ve başlangıç ve bitiş noktalarını bulur.
- **loadMap(const std::string& filename)**: Bu işlev, dosyayı okuyarak labirent haritasını `grid` adında bir 2D vektöre yükler.
- **findStartAndEnd()**: Bu işlev, labirentin ilk satırında (0. satır) ve son satırında (19. satır) bulunan boşlukları (başlangıç ve bitiş noktaları) bulur.
- **solveMaze()**: Bu işlev, bir yığın (stack) kullanarak labirenti çözmeye çalışır. Mevcut konumu yığından çıkarır, hareket eder ve çözümü bulana kadar devam eder. Eğer bir çıkmaza girerse, geri döner ve diğer yönleri dener.
- **nextPosition(const Position& pos, Direction dir) const**: Bu işlev, belirtilen yön doğrultusunda bir sonraki konumu döner.
- **isValid(const Position& pos) const**: Bu işlev, verilen konumun geçerli olup olmadığını kontrol eder (labirentin sınırları içinde ve geçilebilir bir hücre).

- **displayMaze() const**: Bu işlev, mevcut labirent haritasını ekranda gösterir ve terminali temizler.
- **showSolution()**: Bu işlev, başlangıç ve bitiş noktalarını işaretler ve `solveMaze()` işlevini çağırarak çözümü ekranda gösterir.

```
#include "Konum.h"

Position Position::move(Direction direction) const {
    switch (direction) {
        case DOWN: return Position(row + 1, col, DOWN);
        case LEFT: return Position(row, col - 1, LEFT);
        case UP: return Position(row - 1, col, UP);
        case RIGHT: return Position(row, col + 1, RIGHT);
        default: return *this;
    }
}

Position Position::oppositeDirection() const {
    switch (dir) {
        case DOWN: return Position(row, col, UP);
        case UP: return Position(row, col, DOWN);
        case LEFT: return Position(row, col, RIGHT);
        case RIGHT: return Position(row, col, LEFT);
        default: return *this;
```

Amaç: Konum sınıfı, labirent içindeki bir hücrenin konumunu ve yönünü temsil eder.

İşlevler:

move(Direction direction): Bu işlev, mevcut konumdan belirli bir yönde (DOWN, LEFT, UP, RIGHT) hareket eden yeni bir Position nesnesi döner.

oppositeDirection(): Bu işlev, mevcut yönün ters yönünü döner. Bu, geri izleme için kullanılır.

operator== ve operator!=: Bu operatörler, iki Position nesnesinin eşitliğini veya eşitsizliğini kontrol eder.