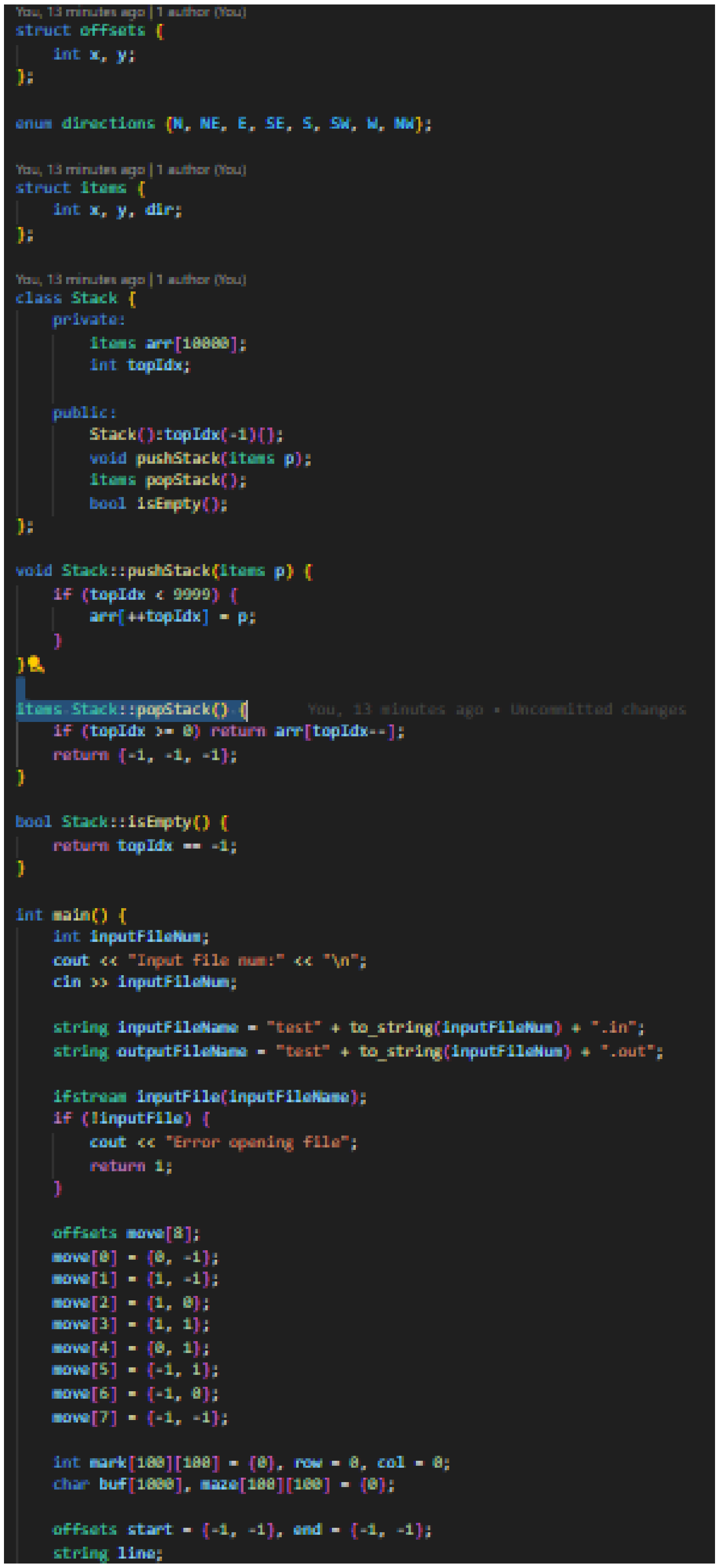
2025 Data Structure HW3 – Maze



```
You, 13 minutes ago | 1 author (You)
struct offsets {
    int x, y;
};

enum directions {N, NE, E, SE, S, SW, W, NW};

You, 13 minutes ago | 1 author (You)
struct items {
    int x, y, dir;
};

You, 13 minutes ago | 1 author (You)
class Stack {
    private:
        items arr[10000];
        int topIdx;

    public:
        Stack():topIdx(-1){};
        void pushStack(items p);
        items popStack();
        bool isEmpty();
};

void Stack::pushStack(items p) {
    if (topIdx < 9999) {
        arr[++topIdx] = p;
    }
}

items Stack::popStack() {        You, 13 minutes ago • Uncommitted changes
    if (topIdx >= 0) return arr[topIdx--];
    return {-1, -1, -1};
}

bool Stack::isEmpty() {
    return topIdx == -1;
}

int main() {
    int inputFileNum;
    cout << "Input file num:" << "\n";
    cin >> inputFileNum;

    string inputFileName = "test" + to_string(inputFileNum) + ".in";
    string outputFileName = "test" + to_string(inputFileNum) + ".out";

    ifstream inputFile(inputFileName);
    if (!inputFile) {
        cout << "Error opening file";
        return 1;
    }

    offsets move[8];
    move[0] = {0, -1};
    move[1] = {1, -1};
    move[2] = {1, 0};
    move[3] = {1, 1};
    move[4] = {0, 1};
    move[5] = {-1, 1};
    move[6] = {-1, 0};
    move[7] = {-1, -1};

    int mark[100][100] = {0}, row = 0, col = 0;
    char buf[1000], maze[100][100] = {0};

    offsets start = {-1, -1}, end = {-1, -1};
    string line;
```

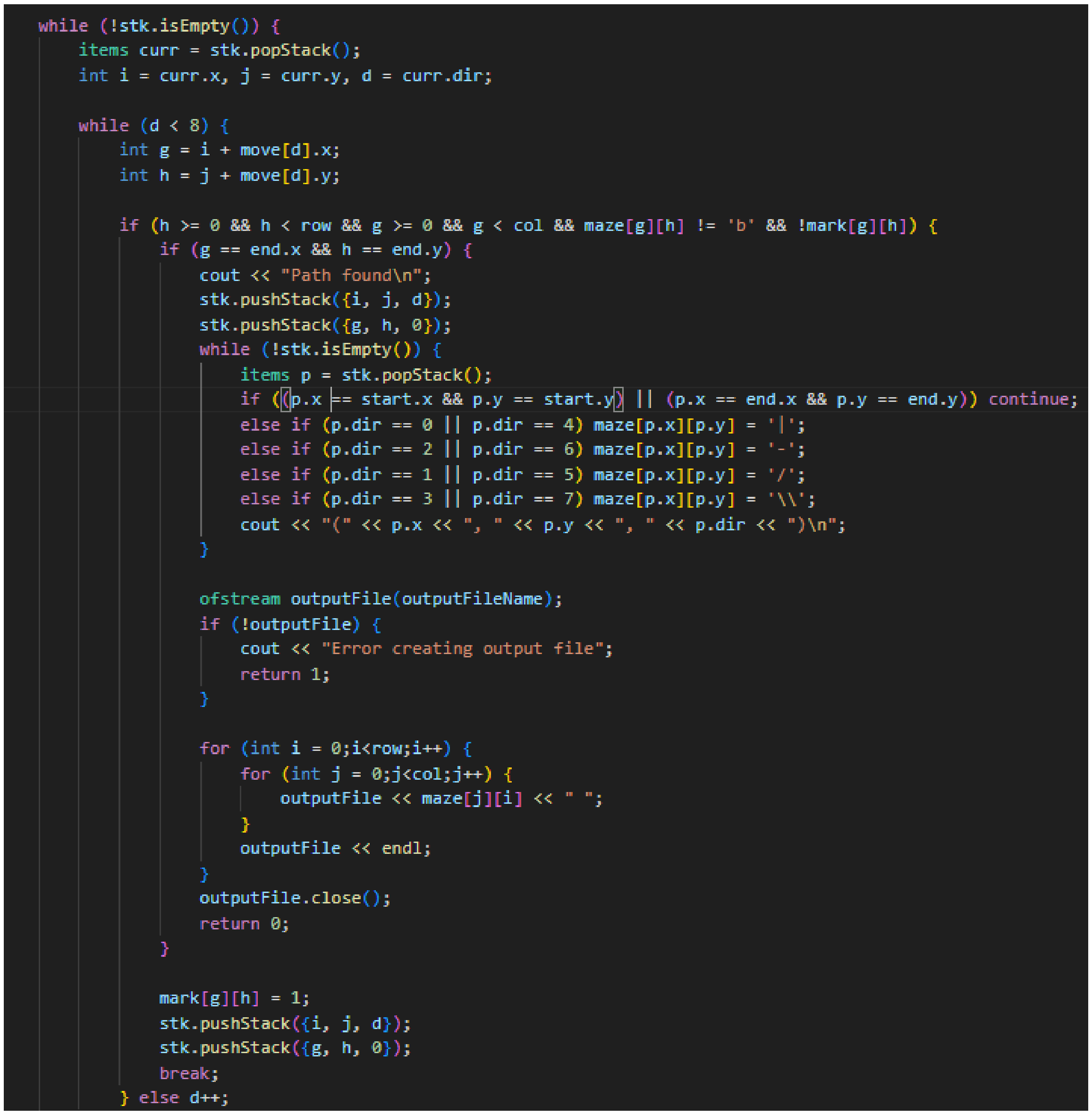
An “offset“ structure to store coordinates.

An “item” structure to store x, y and direction in stack.

The Stack class implementing push, pop, isEmpty.

Reading from the .in file and storing it as a 2d array

Storing 8 possible directions in the move[8] array



```cpp
while (!stk.isEmpty()) {
    items curr = stk.popStack();
    int i = curr.x, j = curr.y, d = curr.dir;

    while (d < 8) {
        int g = i + move[d].x;
        int h = j + move[d].y;

        if (h >= 0 && h < row && g >= 0 && g < col && maze[g][h] != 'b' && !mark[g][h]) {
            if (g == end.x && h == end.y) {
                cout << "Path found\n";
                stk.pushStack({i, j, d});
                stk.pushStack({g, h, 0});
                while (!stk.isEmpty()) {
                    items p = stk.popStack();
                    if ((p.x == start.x && p.y == start.y) || (p.x == end.x && p.y == end.y)) continue;
                    else if (p.dir == 0 || p.dir == 4) maze[p.x][p.y] = '|';
                    else if (p.dir == 2 || p.dir == 6) maze[p.x][p.y] = '-';
                    else if (p.dir == 1 || p.dir == 5) maze[p.x][p.y] = '/';
                    else if (p.dir == 3 || p.dir == 7) maze[p.x][p.y] = '\\';
                    cout << "(" << p.x << ", " << p.y << ", " << p.dir << ")\n";
                }

                ofstream outputFile(outputFileName);
                if (!outputFile) {
                    cout << "Error creating output file";
                    return 1;
                }

                for (int i = 0;i<row;i++) {
                    for (int j = 0;j<col;j++) {
                        outputFile << maze[j][i] << " ";
                    }
                    outputFile << endl;
                }
                outputFile.close();
                return 0;
            }

            mark[g][h] = 1;
            stk.pushStack({i, j, d});
            stk.pushStack({g, h, 0});
            break;
        } else d++;
```

This is the main logic in my program. While the stack isn't empty, for each loop it'll store its current x coordinates & y coordinates and try through 8 directions N, NE, E, SE, S, SW, W, NW and store the current coordinates with the next heading in the stack. And it'll repeat the loop until the end is reached also marking each visited path, once it meets the end, it'll start writing the .out file by popping from the stack and assign -, |, \ accordingly to the maze array, then write it to test.out.

Conclusion

This program still has some imperfections such as not being able to search out the shortest path from start to end or routing expensive routes due to the directions being tried are in a strict order, but still it’s a great problem to learn and understand the implementations of the stack data structure.