Robot Movement

Problem

You need to implement a robot that moves and draws on an **N × N** square grid by executing a series of commands. The robot follows these movement and drawing rules:

1. **Grid Definition:**
   The grid is defined as an **N × N** square with coordinates ranging from **(0,0)** to **(N-1, N-1)**. The **origin (0,0) is at the top-left corner**.

2. **Commands:**

   - **DIMENSION N**
     In text format: "DIMENSION[space]N"
     Sets the grid size to **N × N**, **N** is a positive integer.

   - **MOVE_TO x, y**
     In text format: "MOVE_TO[space]x[,]y"
     Moves the robot to **(x, y) without** drawing. **x** and **y** are constrained in range **[0, N-1]**.

   - **LINE_TO x, y**
     In text format: "LINE_TO[space]x[,]y"
     Moves the robot to **(x, y)** while drawing a line on the grid. **x** and **y** are constrained in range **[0, N-1]**.

3. **Drawing Behavior:**

   - The robot is always located to the **center of a cell** when given a coordinate.

   - When moving with **LINE_TO**, it **draws a Bresenham line**, marking all cells that the line passes through with a "+" symbol.

   - The initial position of the robot is always **(0, 0)**.

Your Task

Write a **console program** that:

- Reads commands from a **text file**.
- Print the **final grid** to the console. **After processing all the commands**, any unvisited cell should be represented by a "." symbol

Example

Input Commands (excluding triple-quoted symbols)

```
“””
DIMENSION 5

MOVE_TO 1,1

LINE_TO 3,3

LINE_TO 3,2
“””
```

Expected Output (excluding triple-quoted symbols)

```
“””
. . . . .

. + . . .

. . ++ .

. . . + .

. . . . .
“””
```

Explanation

x y	0	1	2	3	4
0	.	.	.	.	.
1	.	+	.	.	.
2	.	.	+	+	.
3	.	.	.	+	.
4	.	.	.	.	.

1. The robot starts at (0,0)
2. MOVE_TO 1,1 move the robot to (1,1) without drawing
3. LINE_TO 3,3 draws a **Bresenham line**, marking: (1,1) → (2,2) → (3,3)
4. LINE_TO 3,2 moves to (3,2), marking that cell.

Requirements

Bare minimum

1. **The program should work as expected**

2. **Pure C++**

   - No external libraries are allowed **in the program**

   - Use only standard C++

3. **Object-Oriented Design (OOP)**
   Design your solution in a way that allows scalability (e.g., adding new commands like CIRCLE_TO or exporting the output as an image in the future).

Things that can prove your skillset for us

1. You **should submit only code-related work**. Any unrelated content in your submission will never be considered as a good thing.
2. Modern C++ is always welcome
3. **Documentation:** for building the project, for logic in the codes, for your approach, etc. We can build your project by ourselves, but **it is always good** for writing build instructions.
4. **Error handling**: when parsing command, input validation, etc.
5. **Safety first:** memory management, memory leak, writing test cases, etc.
   You **can use** third party testing libraries
6. **Using battle-tested build systems:** CMake, gn, cross-platform support, etc.
7. **Performance-awareness**
   what if the grid size is so large?
   what if there are a lot of commands?
   Etc.