Week 1: Fundamentals of Algorithms

COMP0209 Data Structures and Algorithms

Daniel Tozadore (d.tozadore@ucl.ac.uk)

What is an Algorithm?

What is an Algorithm?

How to prepare a birthday cake?

- Mix the ingredients
- Bake
- Topping
- Putting the candles

What is an Algorithm?

How to prepare a birthday cake?

- Mix the ingredients
- Bake
- Topping
- Putting the candles

Objects:
- Flour
- Eggs
- Milk
- Oil
- Choc
- Baking Powder
- Sugar

Procedures
- Mix the dry ingredients in a bow
- Break the eggs and stir with the milk and oil
- …

What is an Algorithm?

"A finite, well-defined, step-by-step method that contains inputs and outputs and guarantees the solution to a problem when followed strictly"

"Introduction to Algorithms" by Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest and Clifford Stein (MIT press)

Is every program an Algorithm?

Coding Sequences

Algorithm — a finite, well-defined, step-by-step method that contains inputs and outputs and guarantees the solution to a problem when followed strictly. [Sources: CLRS (MIT Press); Britannica]

Procedure/Subroutine; Function/Method — a callable unit of code. Many languages distinguish procedures (no return value) from functions (return a value); in OO contexts, class-bound functions are methods. [Sources: Wikipedia (Function in programming); Loyola Marymount University notes]

Process (Operating System) — an instance of a running program with its own memory and state, managed by the OS (PID, scheduling, lifecycle). [Source: Baeldung — Process Lifecycle]

Software Process / SDLC — a structured methodology for planning, building, testing, deploying, and maintaining software (e.g., waterfall, agile). [Sources: IBM SDLC; Microsoft Power Platform SDLC overview]

Pseudocode — a language-agnostic, human-readable description of an algorithm using programming-like constructs, intended for design and communication rather than execution. [Sources: Wikipedia — Pseudocode; GeeksforGeeks tutorial]

Flowchart — a graphical representation of control flow (start/end, process, decision, I/O) using standardized symbols; ISO 5807 defines symbols and conventions. [Sources: ISO 5807; Venngage symbols guide]

Heuristic — a problem-solving technique that trades optimality or completeness for speed, aiming for "good enough" solutions efficiently (e.g., A* with admissible heuristic). [Sources: Wikipedia — Heuristic (CS); Baeldung — Heuristic function]

The “bad programmer” paradox

The “bad programmer” paradox

What type of logic is this?

The “bad programmer” paradox

Jet
@jetblack72

Programmer goes to shop for groceries, wife tells him: "Get a gallon of milk. If they have eggs, get a dozen." So he comes back home **with a gallon of milk and a dozen eggs because he's a programmer, not a poorly programmed computer.**

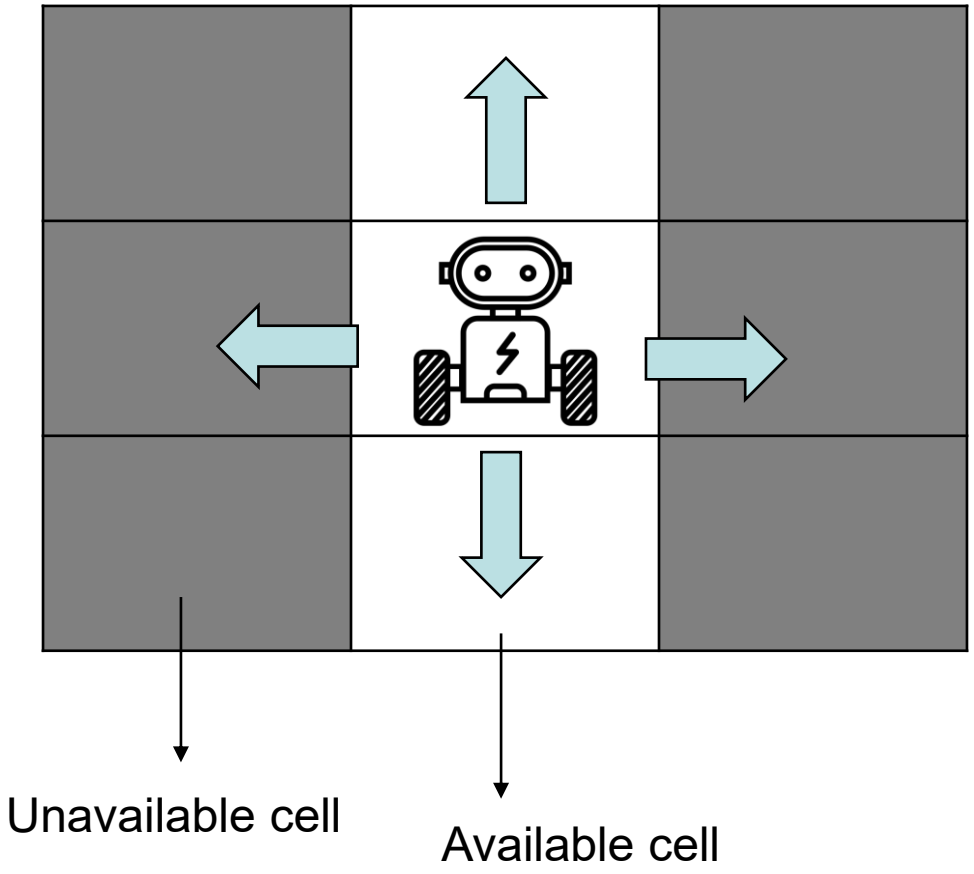
Before we start…

Exercise 1

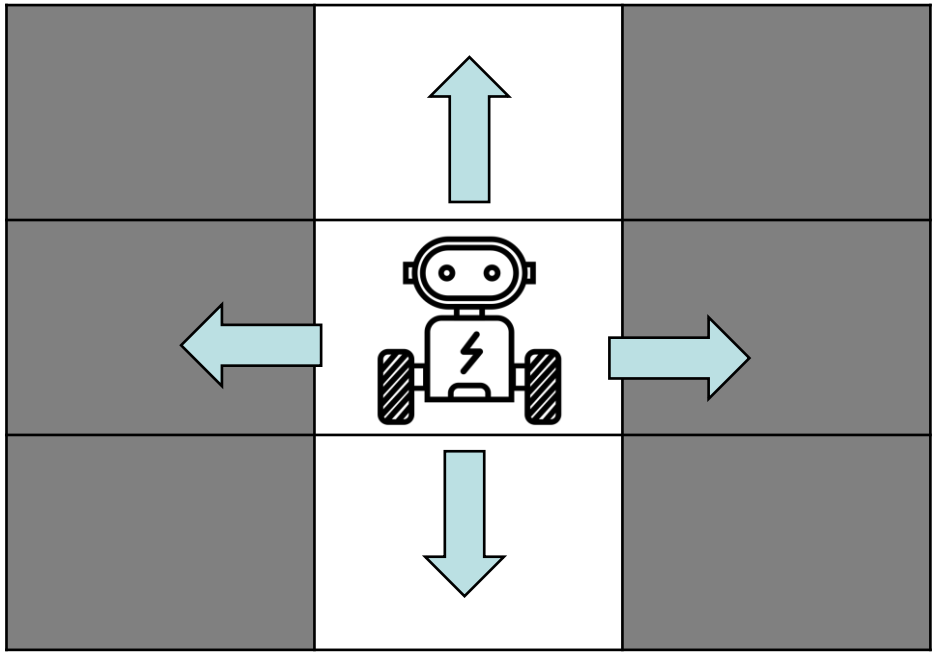
You will have 3 minutes to design an algorithm to make the robot scape this maze

Robot's move:

- Up
- Down
- Right
- Left



Exercise 1



Solutions Raise your hands if you were part of:

1 – Lazy ones:
Move(up)
Move(up)

2 – Used Conditions

3 – Used Loops

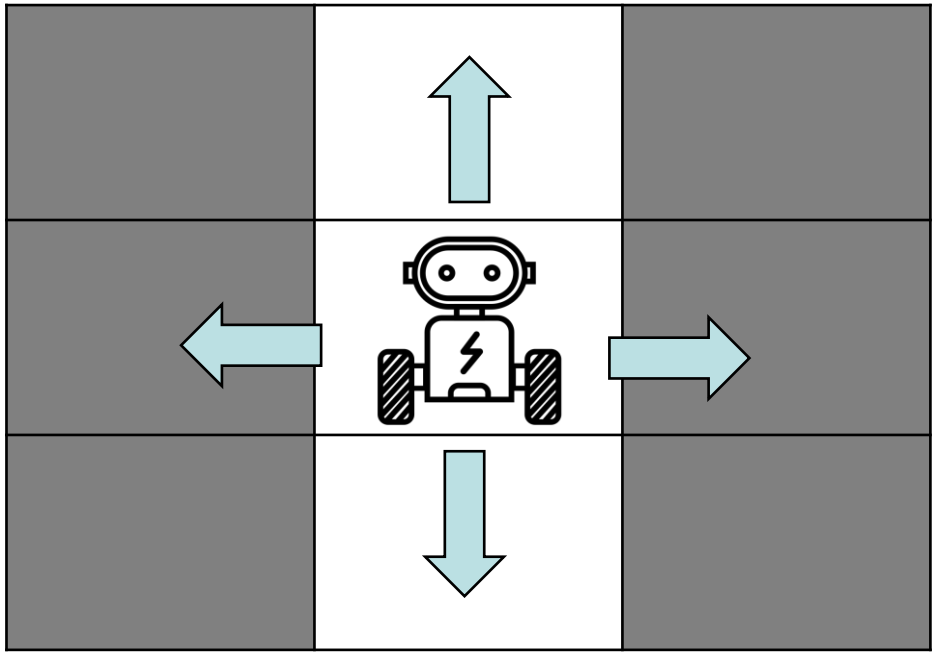
4 - Matrix

5 – Used recursion

6 – Data Structures

7 – Any other technique

Exercise 1



New term resolution:

Let’s work together so we can make it in the end of this module!

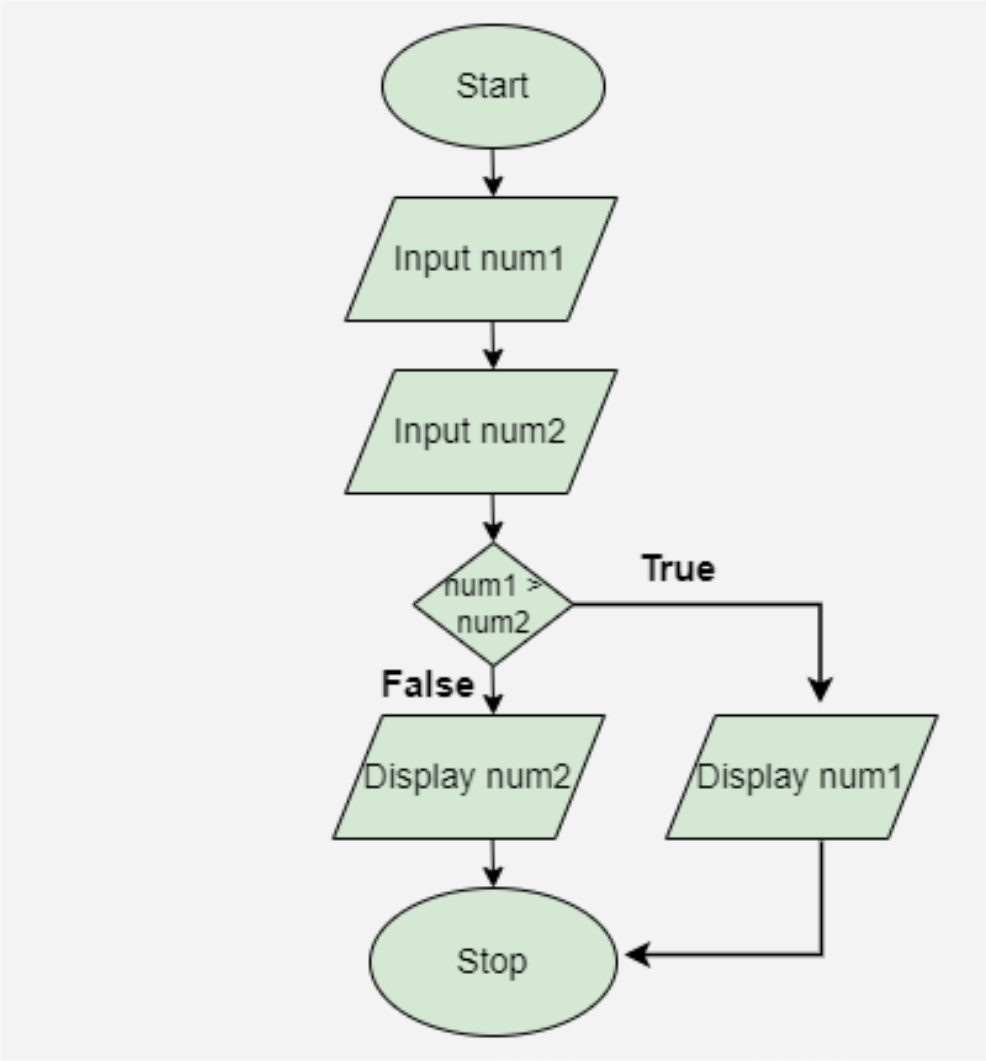
Exercise 1

New term resolution:

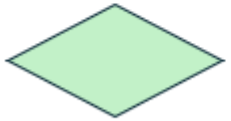
Let’s work together so we can make it in the end of this module!

Flowchart

- **Definition:** A **graphical representation** of control flow (start/end, processes, decisions, I/O) using standardized symbols.
- **Purpose:** provide clarity and simplification to complex processes and algorithms.



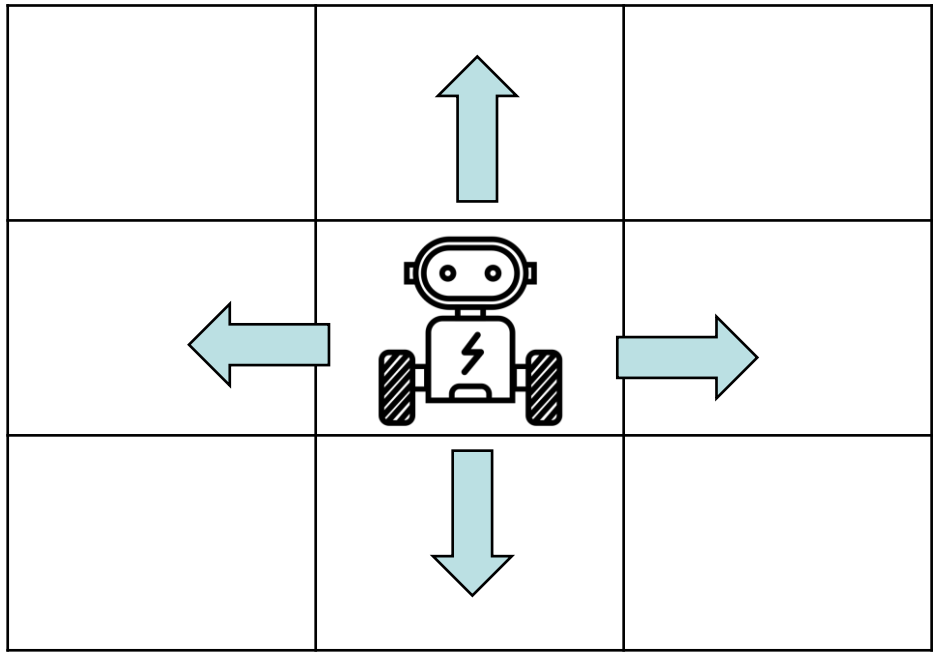
Flowchart

Symbol	Name	Function
	Oval	Represents the start or end of a process
	Rectangle	Denotes a process or operation step
	Arrow	Indicates the flow between steps
	Diamond	Signifies a point requiring a yes/no
	Parallelogram	Used for input or output operations

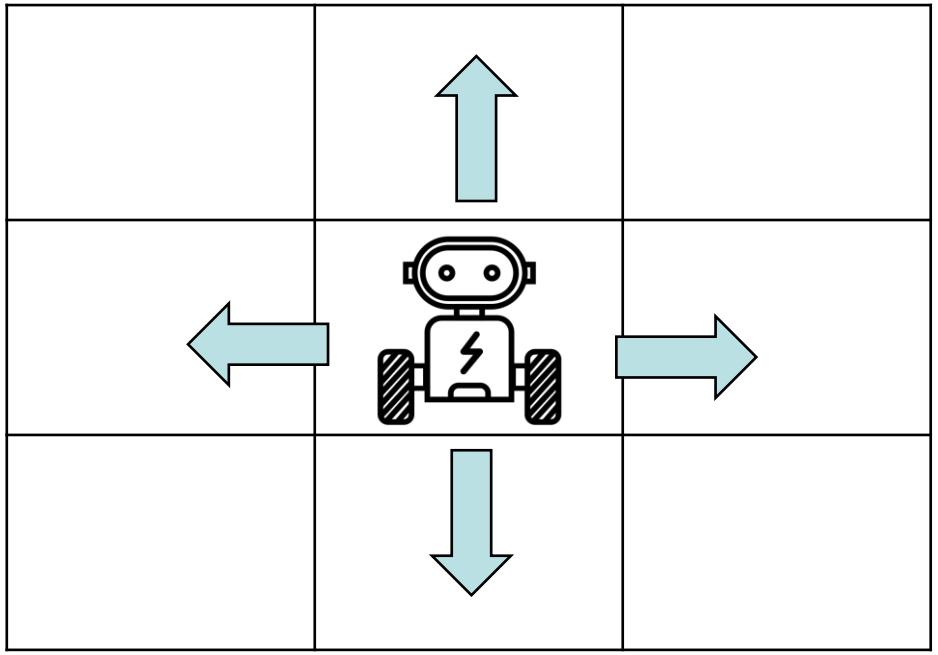
Flowchart example

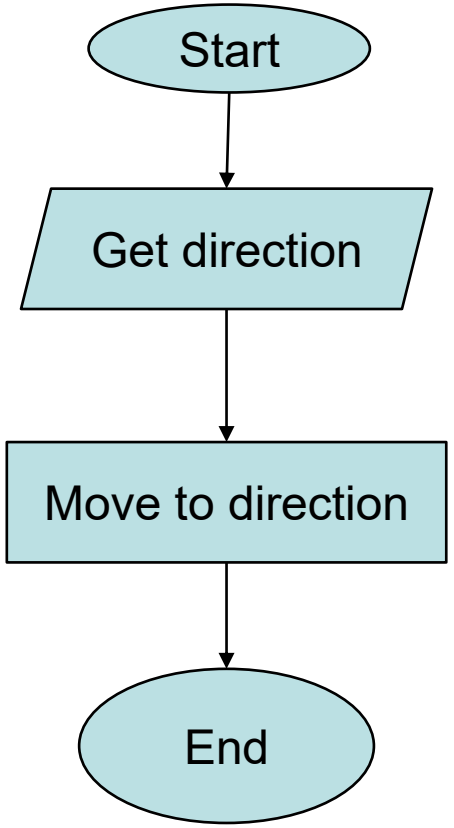
Robot's move:

- Up
- Down
- Right
- Left



Flowchart example

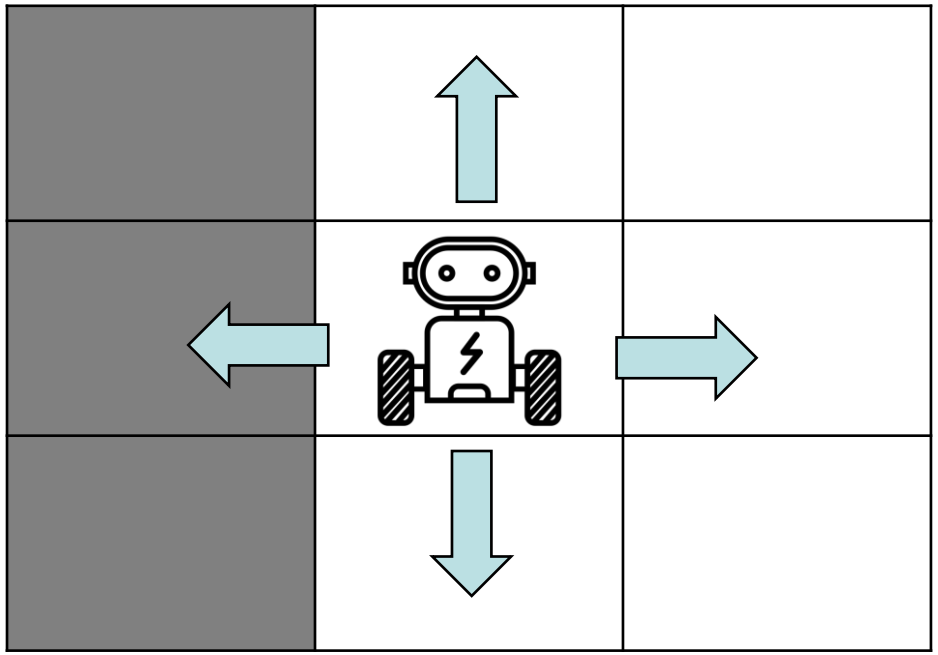


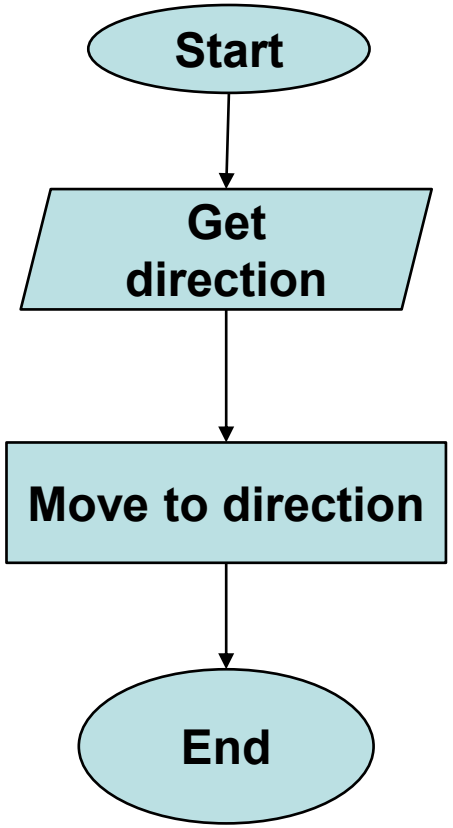


Flowchart: exercise

Now we have obstacles (dark cells).

Update the flowchart to check if the cell is free before moving.





Pseudocode

- **Definition:** A language-independent way to describe algorithms using structured steps.
- **Purpose:** Focus on logic, not syntax.

How to (pseudo)code:

- Start with the goal of the algorithm as a comment
- Use code ident for the blocks
- Use “end” and the command to finish a block
- Test your code for failures

```
// Recursively calculate the factorial of n

Algorithm Factorial(n)
    if n = 0 then
        return 1
    else
        return n * Factorial(n - 1)
    end if
end algorithm
```

Pseudocode

Why to Use?

- Language-independent
- Improves clarity
- Easier debugging
- Facilitates communication
- Serves as a blueprint for coding

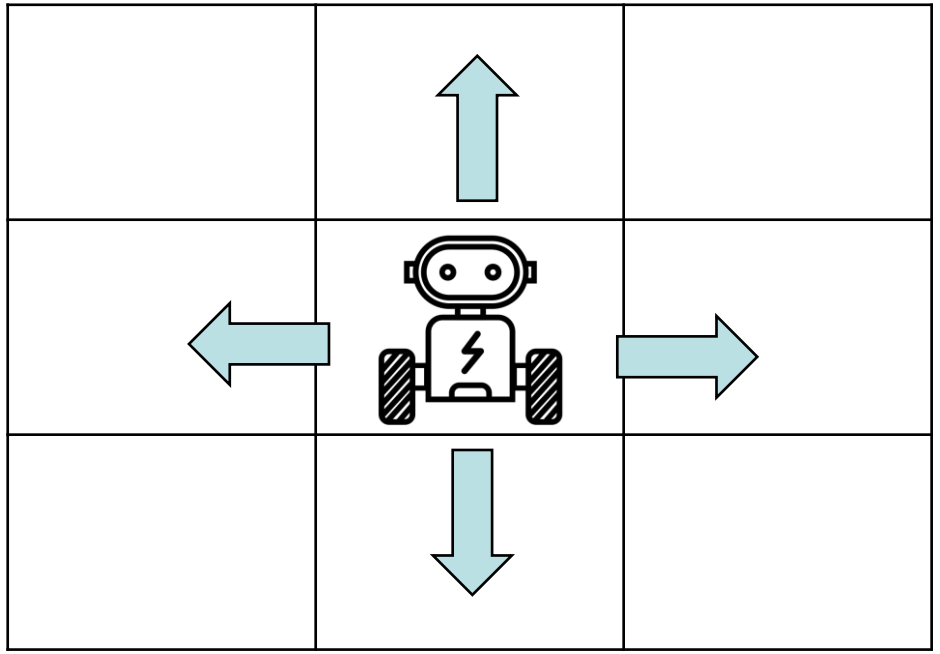
Pseudocode

Token	Meaning	Example	Notes
←	Assignment (store right side into left side)	sum ← 0	Text alternatives: <- or :=
=	Equality comparison	IF x = 10 THEN	Distinct from assignment
≠	Not equal	IF x ≠ y THEN	Alternative: !=
<, ≤, >, ≥	Comparisons	IF a ≤ b THEN	Standard math comparisons
AND, OR, NOT	Logical operators	IF a > 0 AND b > 0 THEN	Short-circuit logic as usual
[]	Indexing / list literal	A[0], A ← [1,2,3]	0-based indexing in this handout
{}	Set or map literal	S ← {1,2,3} / M ← {k: v}	Context defines set vs map
…	Ellipsis (omitted steps)	// …	Use sparingly
:	Parameter labels / map keys	{x: 1, y: 2}	Also used in messages
NIL	No value / null	IF node = NIL THEN …	For absence
//	Comment	// explain logic here	Everything after // is ignored

Pseudocode example

Robot's move:

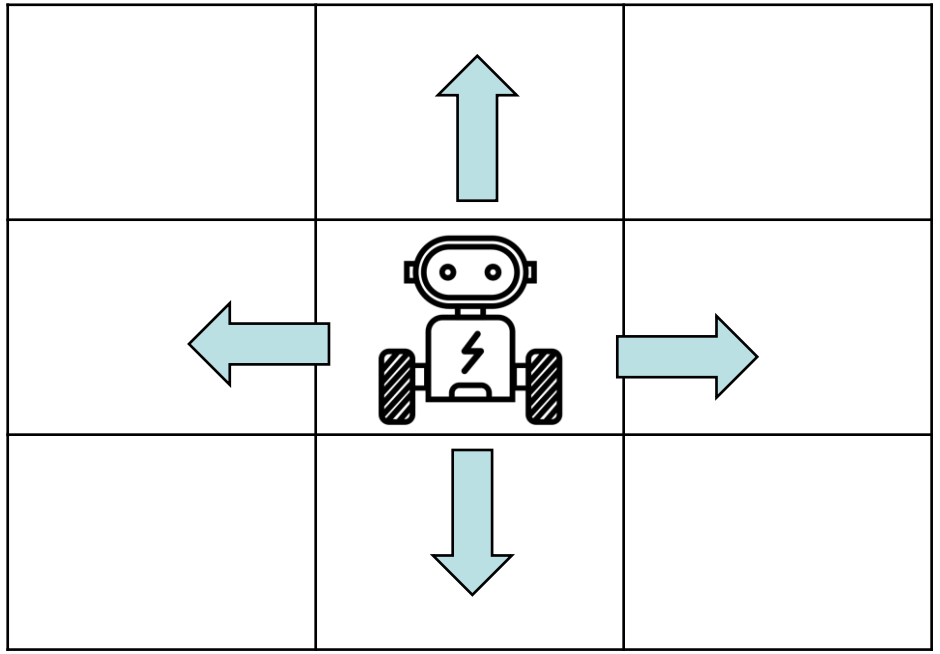
- Up
- Down
- Right
- Left



Pseudocode example

Robot's move:

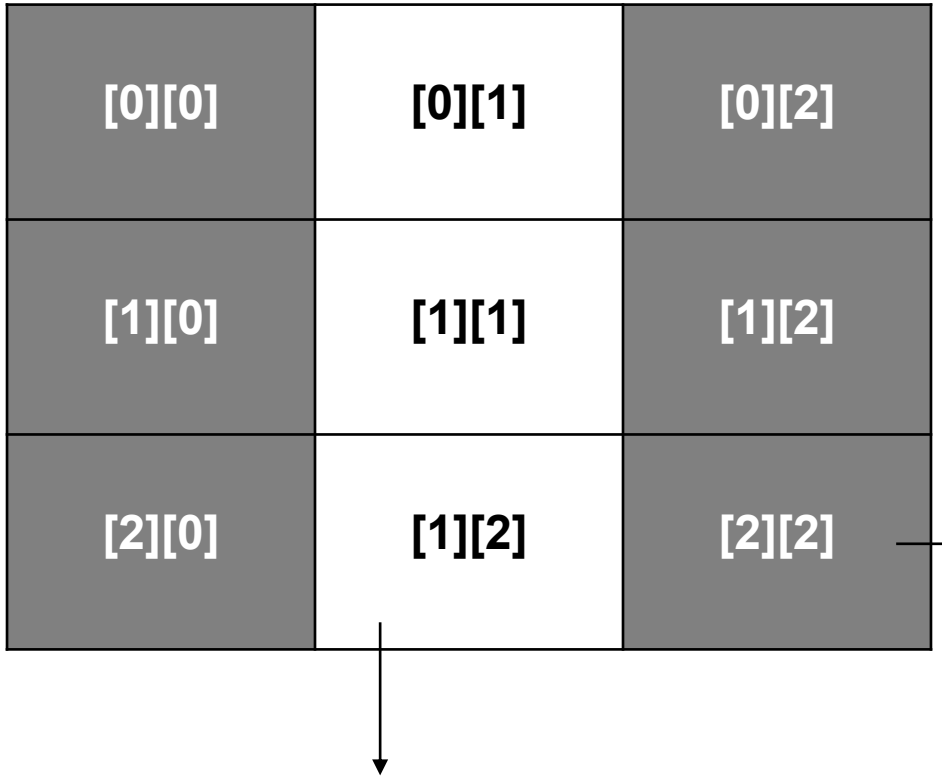
- Up
- Down
- Right
- Left



Question:
Does high level descriptions work when translating to actual code?

Scenario

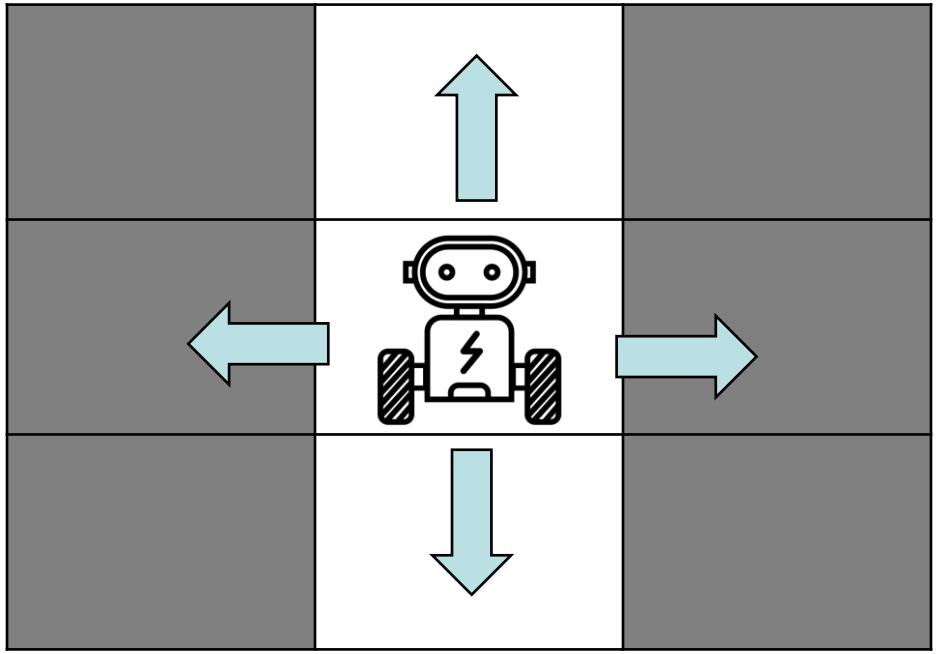
Map:3x3 Grid



Scenario

Robot's move:

- Up
- Down
- Right
- Left



robot position = [1][1]

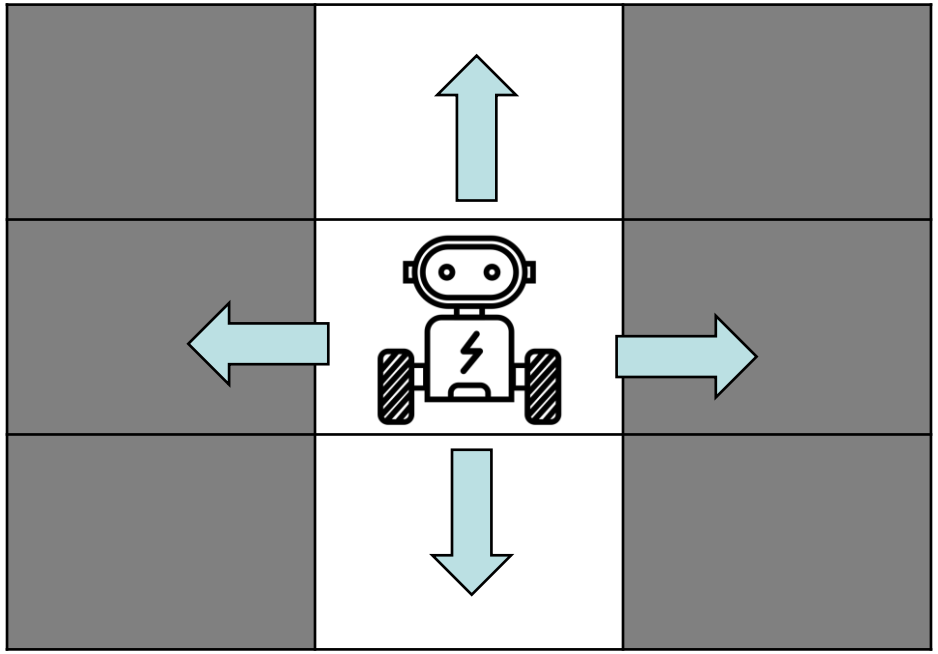
move(down)

robot position = ?

Scenario

Robot's move:

- Up
- Down
- Right
- Left



robot position = [1][1]

move(down)

robot position = [2][1]

Let's call it *robot_position.x* and *robot_position.y*

Pseudocode exercise/example:

- *robot_position*{robot_position.x, robot_position.y} -> SET

Implement the function “move(direction, robot_position)” that will return the updated position of the robot (robot_position.x, robot_position.y) given a direction (up,down,right, left) and the current position.

Pseudocode exercise/example:

Implement the function "move(direction, robot_position)" that will return the updated position of the robot (robot_position.x, robot_position.y) given a direction (up,down,right, left) and the current position.

```
// calculate the new position based on current
position and direction to move

Move (direction, robot position)
    if direction = "up"
        robot_position.x ← robot_position.x - 1
        return robot_position

    else if direction = "down"
        robot_position.x ← robot_position.x + 1
        return robot_position

    else if direction = "right"
        robot_position.y ← robot_position.y + 1
        return robot_position

    else if direction = "left"
        robot_position.y ← robot_position.y - 1
        return robot_position

    end if
end algorithm
```

Programming in Pairs

Using the card decks

- Working in pairs, you will write algorithms to manipulate your subset of cards
- Both peers write the pseudocode, then swap algorithms and debug your peer's code

- Available commands:
  - Check value
  - Write down value

n	i	x	…
7	0	8	
7	1	3	
7	2	4	

Exercises

Follow this part in your exercise sheet!

And mind your own sheet!

Exercises

Given a set S of integers (Your cards):

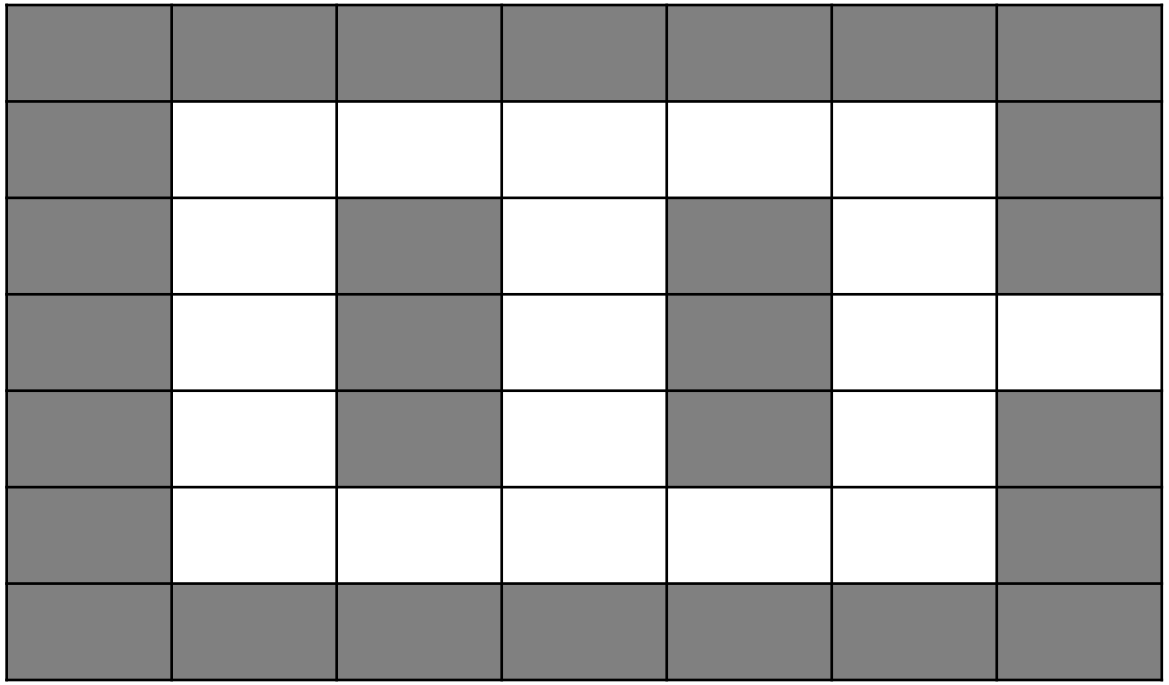
1. Implement an algorithm to return the sum of the values in your set.
2. Implement an algorithm to count how many odd (even) numbers are in your set
3. Implement an algorithm to return the max/min number in your set
4. Implement an algorithm to return the 2 smallest numbers in a given set
5. Implement an algorithm to return the set sorted (ascending/descending)

Programming hacks

- Apply your performance measurement function to the implemented algorithms. What were their final values?
- Debugging edge cases?

For next session:

- Implement your best and complete pseudocode to get your robot out of a maze (mesh)



Your attention was all I needed ☺

Please feedback me at:

- COMP0209 DSA - Feedback

Or by the QR code: