

Unit 5 - Lesson 3

Row-Major Traversal



Warm Up



 **Do This:**

Write an algorithm to find the values **greater than 10** in this 2D array.

6	3	12
5	10	2
9	4	8
1	11	7



Activity



Lesson Objectives

By the end of this lesson, you will be able to . . .

- Trace code segments that traverse a two-dimensional (2D) array in row-major order
- Write nested loops to traverse a two-dimensional (2D) array in row-major order
- Apply standard one-dimensional (1D) array algorithms to a two-dimensional (2D) array



Question of the Day

How can I traverse a 2D array?



Predict and Run

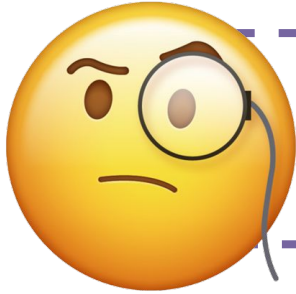


Navigate to Lesson 3, Level 1



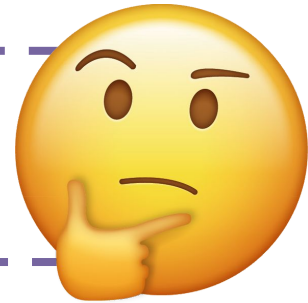
Do This:

1. Predict the output of the program
There are no wrong answers!
2. Run it to compare your prediction with the results




What did you notice about the code in this program?

What do you wonder about the code in this program?



Row-Major Traversal

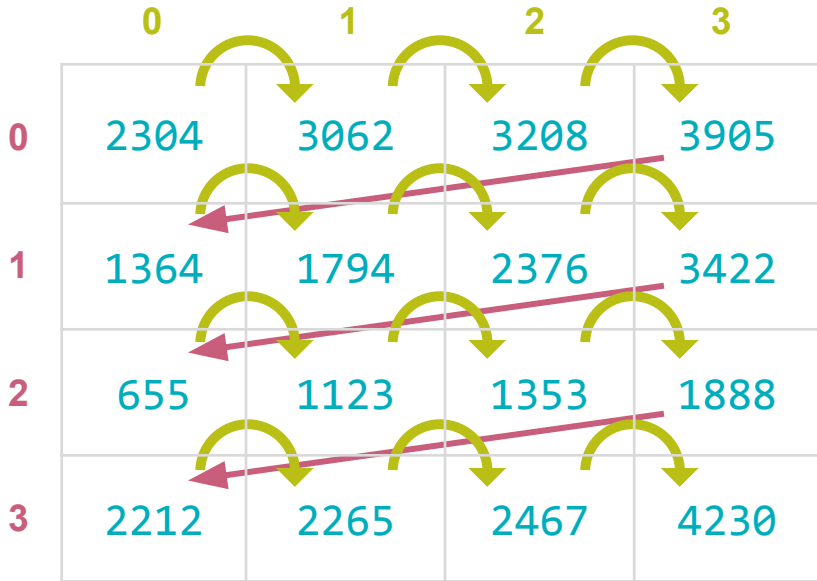
How are nested loops used to traverse a 2D array?

Complete the guided notes on the  **Unit 5 Guide**.





When we access data by traversing each full row from top to bottom, we are traversing in **row-major order**.



```
for (int row = 0; row < values.length; row++) {
    for (int col = 0; col < values[0].length; col++) {
        System.out.println(values[row][col]);
    }
}
```





Tracing 2D Array Traversals

You and your partner should have:

- Tracing 2D Array Traversals activity guide
- pen / pencil



Name(s) _____ Period _____ Date _____

Activity Guide - Tracing 2D Array Traversals

Part I: Draw a Trace Table

Draw a trace table that has:

- a column labeled **Line** for the line number
- a column labeled with the variable name for each variable declared, including loop control variables
- a column labeled **Output** for values printed

Part II: Complete the Trace Table

Trace the code segment. For each step, create a new row in your trace table and fill it in with the following:

- Write the line number you are currently executing in the **Line** column.
- Write the current value of the loop control variable in the loop control variable's column.
- If a variable was assigned a value, write the new value in the corresponding column.
- Fill in the other blank cells with the value from the cell located in the row above it.
- If a variable has not been created yet, leave the cell blank.

1



Do This:

Complete the **Tracing 2D Array Traversals** activity guide.



Name(s) _____ Period _____ Date _____

Activity Guide - Tracing 2D Array Traversals

C
O
D
E

Part I: Draw a Trace Table

Draw a trace table that has:

- a column labeled **Line** for the line number
- a column labeled with the variable name for each variable declared, including loop control variables
- a column labeled **Output** for values printed

Part II: Complete the Trace Table

Trace the code segment. For each step, create a new row in your trace table and fill it in with the following:

- Write the line number you are currently executing in the **Line** column.
- Write the current value of the loop control variable in the loop control variable's column.
- If a variable was assigned a value, write the new value in the corresponding column.
- Fill in the other blank cells with the value from the cell located in the row above it.
- If a variable has not been created yet, leave the cell blank.

1



 **Discuss:**

What are some **questions** we could **answer** by **traversing** through this data?

			
	25	17	22
	18	12	15
	21	19	27
	30	10	23



Writing Algorithms with 2D Arrays

You and your partner should have:

- Writing Algorithms with 2D Arrays activity guide
- pen / pencil



Name(s) _____ Period _____ Date _____

Activity Guide - Writing Algorithms with 2D Arrays

C
O
D
E

Algorithms and the Problem-Solving Process

The **Problem Solving Process** is useful when planning and writing algorithms. This process will help you clarify and break down a problem into manageable steps so you can easily identify the code you need to write for each step.

Define

- Read the instructions carefully to make sure you understand the goals.
- Rephrase the problem in your own words.
- Identify any new skills you are being asked to apply.
- If there is starter code, read it to understand what it does.

Prepare

- Write out or draw the steps you need to take to solve the problem.
- List what you already know how to do and what you don't yet.
- Explain your algorithm to a classmate.
- Review similar programs that you've written in the past.

Try

- Write one small piece at a time.
- Test your program often.
- Use comments to document what your code does.
- Go back to a previous step if you get stuck or don't know whether you've solved the problem.

Reflect

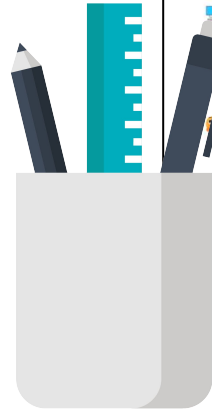
- Compare your program to the defined problem to make sure you've solved all aspects of the problem.
- Ask a classmate to try your program and note places where they struggle or show confusion.
- Ask a classmate to read your code to make sure that your documentation is clear and accurate.
- Try to "break" your program to find types of interaction or input that you could handle better.
- Identify changes or improvements you can make next to your program.

1



✓ Do This:

Choose a problem and **write pseudocode** to find the **sum of all the values** in a **2D array**.



Name(s) _____ Period _____ Date _____

CSA
DE

Activity Guide - Writing Algorithms with 2D Arrays

Algorithms and the Problem-Solving Process

The **Problem Solving Process** is useful when planning and writing algorithms. This process will help you clarify and break down a problem into manageable steps so you can easily identify the code you need to write for each step.

Define

- Read the instructions carefully to make sure you understand the goals.
- Rephrase the problem in your own words.
- Identify any new skills you are being asked to apply.
- If there is starter code, read it to understand what it does.

Prepare

- Write out or draw the steps you need to take to solve the problem.
- List what you already know how to do and what you don't yet.
- Explain your algorithm to a classmate.
- Review similar programs that you've written in the past.

Try

- Write one small piece at a time.
- Test your program often.
- Use comments to document what your code does.
- Go back to a previous step if you get stuck or don't know whether you've solved the problem.

Reflect

- Compare your program to the defined problem to make sure you've solved all aspects of the problem.
- Ask a classmate to try your program and note places where they struggle or show confusion.
- Ask a classmate to read your code to make sure that your documentation is clear and accurate.
- Try to "break" your program to find types of interaction or input that you could handle better.
- Identify changes or improvements you can make next to your program.



Practice



Navigate to Lesson 3, Level 2



Do This:

1. **Level 2** – Check for Understanding
2. **Level 3** – Implement your algorithm to find the sum of all the values in a 2D array
3. **Level 4** – Traverse a 2D array and find the average of all its values or of each row

Wrap Up





 **Discuss:**

How did you use your skills as a software engineer to implement your algorithms?



Problem Solver



Creator



Collaborator



Learner



Today, you learned about . . .

- Tracing code segments that traverse a two-dimensional (2D) array in row-major order
- Writing nested loops to traverse a two-dimensional (2D) array in row-major order
- Applying standard one-dimensional (1D) array algorithms to a two-dimensional (2D) array



Question of the Day

How can I traverse a 2D array?



Key Vocabulary

- **row-major order:** traversing a 2D array by accessing each row from top to bottom