Lec 18 Array 2D:

This lesson explains how to use one-dimensional and multi-dimensional arrays in programming, focusing on two-dimensional arrays (2D arrays). It covers the difference between regular (rectangular) and jagged (irregular) arrays, how to create them, fill them with data, and access their elements using loops. Understanding these concepts helps organize and manage complex data in a structured way.

```
Regular Array: { {1, 2, 3, 4},
                 {5, 6, 7, 8},
                 {9, 4, 5, 2},
               }
Jagged Array: { {1, 2, 3, 4},
                {5, 6, 7},
                {9, 4, 5, 2, 3},
              }
```

Below code will work fine for arr(2d array) if it is regular or jagged:

```
for(int i = 0; i < arr.length; i++){
    for(int j = 0; j < arr[i].length;  j++){
    }
}
```

Recap of One-Dimensional Arrays

A one-dimensional array stores a list of values in a single line, such as roll numbers for students. You can create arrays of basic types (like integers) or objects (like Employee), and fill them either by programmer input or by taking user input during program execution.

Introduction to Multi-Dimensional Arrays

Multi-dimensional arrays, such as 2D or 3D arrays, allow storing data in layers. For example, a 2D array can represent departments and students, where each department contains its own list of student roll numbers.

Understanding the Structure of 2D Arrays

A 2D array is like a table with rows and columns, where data is organized in two layers. This is useful when you need to group related data, such as students within departments.

When to Use 2D Arrays

Use a 2D array when you have data grouped in two levels, like departments and students. Each row can represent a department, and each column a student within that department.

Extending to Higher Dimensions

Arrays can be extended to three or more dimensions, such as colleges containing departments, which in turn contain students. Each added dimension represents another layer of grouping.

Regular vs. Jagged Arrays

Regular arrays (rectangular) have equal-sized rows, while jagged arrays (irregular) allow each row to have a different length. Both can be created by programmer or user input, and are useful for different data structures.

Creating and Accessing 2D Arrays (Programmer Input)

A regular 2D array can be created by directly assigning values in a nested structure. Each row is a one-dimensional array, and all rows have the same length. Accessing elements uses two indices: one for the row (department), one for the column (student).

Example: Departments and Students in 2D Arrays

Three departments each with four students can be stored in a 2D array. The length of the array gives the number of departments, and the length of each sub-array gives the number of students per department.

Array Lengths and Indexing

The length of the main array gives the number of groups (departments), and each sub-array's length gives the count within that group. Indexing is used to access or modify specific data points.

Regular Array Characteristics

In a regular 2D array, all sub-arrays must have the same length, ensuring a consistent structure like a matrix or table.

Printing 2D Array Elements with Loops

Nested loops are used to traverse and print all elements in a 2D array. The outer loop iterates over rows (departments), and the inner loop iterates over columns (students).

Dynamic Access for Jagged Arrays

When sub-arrays have different lengths, use the length property of each sub-array for the inner loop to avoid errors and handle irregular data structures.

Using Enhanced For-Loops

Enhanced for-loops (for-each) can simplify accessing elements in both regular and jagged arrays, making the code cleaner and easier to read.

User Input for 2D Regular Arrays

To create a 2D array with user-defined sizes, use input functions to get the number of departments and students per department, then fill the array using nested loops.

User Input for Jagged Arrays

For jagged arrays, the number of items in each group can be set by the user at runtime. Each sub-array is created with a length specified by the user, allowing for variable group sizes.

Practice Task 1: Diagonal Sum in 2D Arrays

Done: check in Online GDB

A common programming task is to find the sum of the diagonal elements in a square 2D array. The solution should work dynamically for any square size, not just fixed sizes.

Practice Task 2: Sum of Numbers Between Two Inputs

Done: check in Online GDB

Another exercise is to take two numbers from the user, ensure they meet certain constraints, and print the sum of all numbers between them, showing each step in the calculation. This structured summary captures the core concepts and practical applications of one-dimensional and multi-dimensional arrays, with special emphasis on 2D arrays, their types, and how to work with them using both programmer and user input.