**Batch:A2               Roll No.:16010123032**

**Experiment / assignment / tutorial No.4**

**Grade: AA / AB / BB / BC / CC / CD /DD**

**Signature of the Staff In-charge with date**

**TITLE : Multi-dimensional Arrays (Jagged Array)**

**AIM:** Write a program which stores information about n players in a two dimensional array. The array should contain the number of rows equal to the number of players. Each row will have a number of columns equal to the number of matches played by that player which may vary from player to player. The program should display player number (index +1), runs scored in all matches and its batting average as output. (It is expected to assign columns to each row dynamically after getting value from the user.

**Expected OUTCOME of Experiment:**

CO1:Apply the features of object oriented programming languages. (C++ and Java)

CO2:Explore arrays, vectors, classes and objects in C++ and Java

**Books/ Journals/ Websites referred:**

1. E. Balagurusamy, “Programming with Java”, McGraw-Hill.
2. E. Balagurusamy, “Object Oriented Programming with C++”, McGraw-Hill.

**Pre Lab/ Prior Concepts:**

Arrays

**Multi-Dimensional Array**:

|  |  |  |  |
| --- | --- | --- | --- |
| 10 | 12 | 43 | 11 22 |
| 20 | 45 | 56 | 1 33 |
| 30 | 67 | 32 | 14 44 |
| 40 | 12 | 87 | 14 55 |
| 50 | 86 | 66 | 13 66 |

60 53 44 12 11

A multi-dimensional array is one that can hold all the values above. You set them up like this:

**int[ ][ ] numbers = new int[**6**][**5**];**

The first set of square brackets is for the rows and the second set of square brackets is for the columns. In the above line of code, we're telling Java to set up an array with 6 rows and 5 columns.

aryNumbers[0][0] = 10;

aryNumbers[0][1] = 12;

aryNumbers[0][2] = 43;

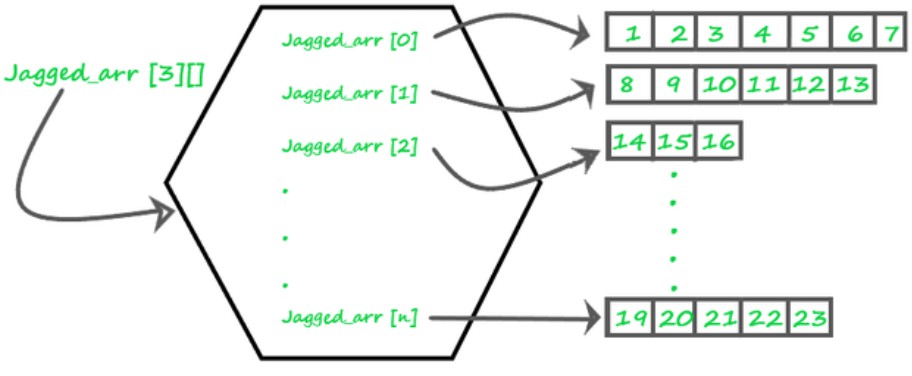
aryNumbers[0][3] = 11;

aryNumbers[0][4] = 22;

So the first row is row 0. The columns then go from 0 to 4, which is 5 items.

**Jagged Array:**

A jagged array, also known as a "ragged array," is an array of arrays where each "inner" array can have different lengths. This contrasts with a rectangular array (or a multi-dimensional array), where every inner array must have the same length. Jagged arrays are useful when dealing with data structures that naturally vary in size, such as lists of lists or matrices with different numbers of columns.



**Theory**

1. **Definition**: A jagged array is an array whose elements are arrays, possibly of different lengths. This means that the length of each inner array can vary.
2. **Memory Layout**: Unlike a rectangular array where memory allocation is continuous, each inner array in a jagged array is a separate array stored at different locations in memory.
3. **Usage**: Jagged arrays are often used in scenarios where the data is inherently irregular. For example, they can be useful in representing data structures like adjacency lists in graphs, where different nodes have different numbers of neighbors.
4. **Advantages**:
   * **Space Efficiency**: Only the required space is allocated for each sub-array, saving memory when dealing with irregular data.
   * **Flexibility**: Allows more flexibility in managing arrays of varying lengths.
5. **Disadvantages**:
   * **Complexity**: Increased complexity in managing and accessing elements.
   * **Performance**: Potentially lower performance due to non-contiguous memory allocation

**Syntax :**

// Declare a jagged array with 3 elements

**int[][] jaggedArray = new int[3][];**

**Class Diagram:**

+ +

| PlayerPerformance |

+ +

| - numPlayers: int |

| - playerRuns: int[][] |

| - numMatches: int |

| - totalRuns: int |

| - battingAverage: double|

| - scanner: Scanner |

+ +

| + main(args: String[]): void |

+ +

**Algorithm:**

1. Start
2. Create Scanner object
3. Read number of players numPlayers from user
4. Initialize a two-dimensional array playerRuns with numPlayers rows
5. For each player i from 0 to numPlayers - 1:
6. Read number of matches numMatches for player i
7. Initialize row playerRuns[i] with size numMatches
8. For each match j from 0 to numMatches - 1:
   1. Read runs scored in match j
   2. Store runs in playerRuns[i][j]
9. For each player i from 0 to numPlayers - 1:
10. Initialize totalRuns to 0
11. For each match j in playerRuns[i]:
    1. Add playerRuns[i][j] to totalRuns
12. Calculate batting average as totalRuns / playerRuns[i].length
13. Print player number, runs scored in all matches, total runs, and batting average
14. Close Scanner object
15. End

**Implementation details:**

import java.util.Scanner;

public class PlayerPerformance {

public static void main(String[] args) { Scanner scanner = new Scanner(System.in);

// Step 3: Get the number of players System.out.print("Enter the number of players: "); int numPlayers = scanner.nextInt();

// Step 4: Create two-dimensional array int[][] playerRuns = new int[numPlayers][];

");

// Step 5: Input data for each player for (int i = 0; i < numPlayers; i++) {

System.out.print("Enter the number of matches played by player " + (i + 1) + ":

int numMatches = scanner.nextInt();

// Initialize the array for the current player playerRuns[i] = new int[numMatches];

System.out.println("Enter the runs scored by player " + (i + 1) + " in each

match:");

for (int j = 0; j < numMatches; j++) { playerRuns[i][j] = scanner.nextInt();

}

}

// Step 6: Display results System.out.println("\nPlayer Performance Report:"); for (int i = 0; i < numPlayers; i++) {

int totalRuns = 0;

// Calculate total runs for the current player for (int j = 0; j < playerRuns[i].length; j++) {

totalRuns += playerRuns[i][j];

}

// Calculate batting average

double battingAverage = (double) totalRuns / playerRuns[i].length;

// Display player information System.out.println("Player " + (i + 1) + ":"); System.out.print("Runs scored in all matches: "); for (int j = 0; j < playerRuns[i].length; j++) {

System.out.print(playerRuns[i][j] + " ");

}

System.out.println(); System.out.println("Total runs: " + totalRuns);

System.out.printf("Batting average: %.2f\n", battingAverage);

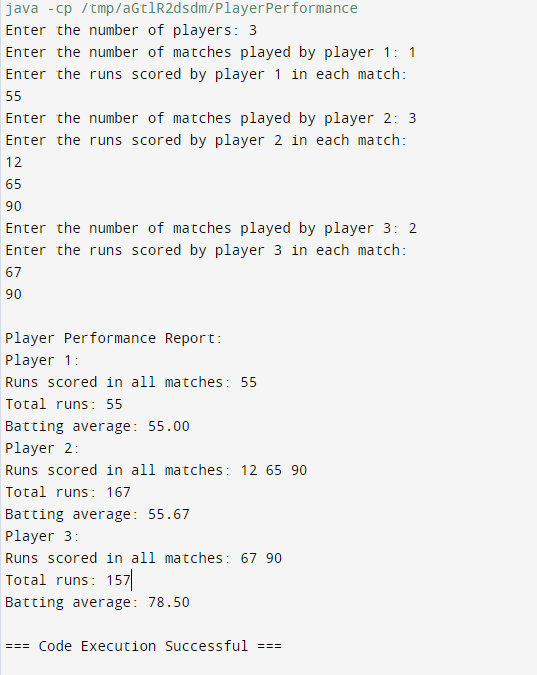
}

// Close the scanner scanner.close();

}

}

**Output:**



**Conclusion:**

This experiment involved studying multidimensional and jagged arrays. It prompts the user to enter the number of players and their respective match scores, calculates the total runs and batting average for each player, and displays the results. The latter was used for implementing the given problem statement and achieving desired results.

We understood the use of arrays, vectors, classes and objects in C++ and Java .

**Date:13/8/24** **Signature of faculty in-charge**

**Post Lab Descriptive Questions:**

**Q.1** Write a program for Given an array arr[] of size N. The task is to find the sum of the contiguous subarray within a arr[] with the largest sum.

import java.util.Scanner;

**public class MaxSubarraySum {**

**public static void main(String[] args) { Scanner scanner = new Scanner(System.in);**

**// Step 1: Take the size of the array from the user System.out.print("Enter the number of elements in the array: "); int N = scanner.nextInt();**

**// Step 2: Initialize the array int[] arr = new int[N];**

**// Step 3: Take array elements from the user System.out.println("Enter " + N + " elements:"); for (int i = 0; i < N; i++) {**

**arr[i] = scanner.nextInt();**

**}**

**// Step 4: Find the maximum sum of the contiguous subarray**

**int maxSum = findMaxSubarraySum(arr);**

**// Step 5: Print the result**

**System.out.println("The sum of the contiguous subarray with the largest sum is: " + maxSum);**

**// Close the scanner scanner.close();**

**}**

**// Function to find the maximum sum of contiguous subarray public static int findMaxSubarraySum(int[] arr) {**

**// Handle empty array case if (arr.length == 0) {**

**throw new IllegalArgumentException("Array should not be empty");**

**}**

**// Initialize variables int maxSoFar = arr[0];**

**int maxEndingHere = arr[0];**

**// Iterate through the array starting from the second element for (int i = 1; i < arr.length; i++) {**

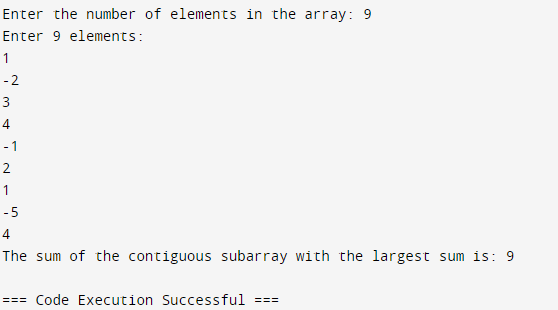
**maxEndingHere = Math.max(arr[i], maxEndingHere + arr[i]); maxSoFar = Math.max(maxSoFar, maxEndingHere);**

**}**

**return maxSoFar;**

**}**

**}**



* 1. Create a jagged array of integers. This array should consist of two 2-D arrays. First 2-D array should contain 3 rows having length of 4,3,and 2 respectively. Second 2-D array should contain 2 rows with length 3 and 4 respectively**.**

**public class JaggedArrayExample { public static void main(String[] args) {**

**// Create the jagged array with two 2-D arrays int[][][] jaggedArray = new int[2][][];**

**// Initialize the first 2-D array with 3 rows of lengths 4, 3, and 2 jaggedArray[0] = new int[3][];**

**jaggedArray[0][0] = new int[4]; // First row with 4 columns jaggedArray[0][1] = new int[3]; // Second row with 3 columns jaggedArray[0][2] = new int[2]; // Third row with 2 columns**

**// Initialize the second 2-D array with 2 rows of lengths 3 and 4 jaggedArray[1] = new int[2][];**

**jaggedArray[1][0] = new int[3]; // First row with 3 columns jaggedArray[1][1] = new int[4]; // Second row with 4 columns**

**// Fill the jagged array with example values**

**fillJaggedArray(jaggedArray);**

**// Print the jagged array printJaggedArray(jaggedArray);**

**}**

**// Method to fill the jagged array with example values public static void fillJaggedArray(int[][][] jaggedArray) {**

**int value = 1; // Start filling with 1 for example for (int i = 0; i < jaggedArray.length; i++) {**

**for (int j = 0; j < jaggedArray[i].length; j++) {**

**for (int k = 0; k < jaggedArray[i][j].length; k++) { jaggedArray[i][j][k] = value++;**

**}**

**}**

**}**

**}**

**// Method to print the jagged array**

**public static void printJaggedArray(int[][][] jaggedArray) { for (int i = 0; i < jaggedArray.length; i++) {**

**System.out.println("2-D Array " + (i + 1) + ":"); for (int j = 0; j < jaggedArray[i].length; j++) {**

**System.out.print("[ ");**

**for (int k = 0; k < jaggedArray[i][j].length; k++) { System.out.print(jaggedArray[i][j][k] + " ");**

**}**

**System.out.print("]");**

**if (j < jaggedArray[i].length - 1) { System.out.print(", ");**

**}**

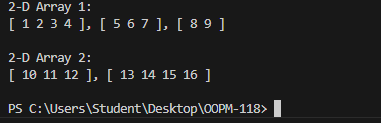
**}**

**System.out.println("\n");**

**}**

**}**

**}**



* 1. **Consider the following code**

int number[] = new int[5];

After execution of this statement, which of the following are true?

1. number[0] is undefined
2. number[5] is undefined
3. number[4] is null
4. number[2] is 0
5. number.length() is 5
   1. (C) & (E)
   2. (A) & (E)
   3. (E)
   4. (B), (D) & (E)

**Ans: The correct answer is (iv)**

* 1. Write a program for Given an array arr[] of size N. The task is to find the sum of the contiguous subarray within a arr[] with the largest sum.

**Same as q1**