Chapter 8: Multidimensional Arrays

Sections 8.1)8.5, 8.8

Textbooks: Y. Daniel Liang, Introduction to Programming with C++, 3rd Edition

These slides were adapted by Prof. Gheith Abandah from the Computer Engineering Department of the University of Jordan for the Course: Computer Skills for Engineers (0907101)
Updated by Dr. Ashraf Suyyagh (Spring 2021)

Outline

- Introduction
- Declaring Two-Dimensional Arrays
- Processing Two-Dimensional Arrays
- Passing Two-Dimensional Arrays to Functions
- Problem: Grading a Multiple-Choice Test
- Multidimensional Arrays

Introduction

Data in a table or a matrix can be represented using a two-dimensional array.

Distance Table (in miles)

	Chicago	*Boston*	*New York*	*Atlanta*	*Miami*	*Dallas*	*Houston*
Chicago	0	983	787	714	1375	967	1087
Boston	983	0	214	1102	1763	1723	1842
New York	787	214	0	888	1549	1548	1627
Atlanta	714	1102	888	0	661	781	810
Miami	1375	1763	1549	661	0	1426	1187
Dallas	967	1723	1548	781	1426	0	239
Houston	1087	1842	1627	810	1187	239	0

Outline

- Introduction
- Declaring Two-Dimensional Arrays
- Processing Two-Dimensional Arrays
- Passing Two-Dimensional Arrays to Functions
- Problem: Grading a Multiple-Choice Test
- Multidimensional Arrays

Declaring Two-Dimensional Arrays

```
elementType arrayName[ROW_SIZE]
[COLUMN_SIZE];
```

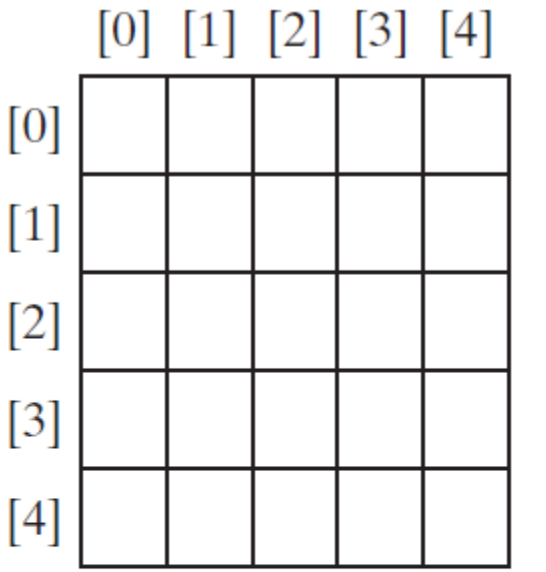
- Example

  ```
  int distances[7][7];
  ```

- An element in a two-dimensional array is accessed through a row and column index.

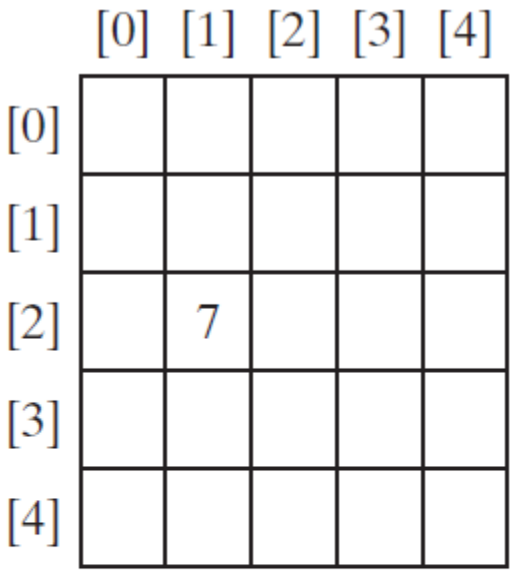
  ```
  int bostonToDalas = distances[1][5];
  ```

Two-Dimensional Array Illustration



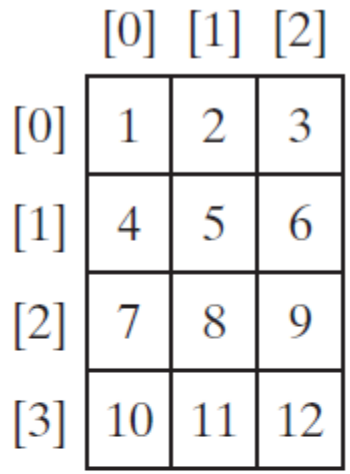

```
int matrix[5][5];
```

(a)




```
matrix[2][1] = 7;
```

(b)




```
int m[4][3] =
  {{1, 2, 3},
   {4, 5, 6},
   {7, 8, 9},
   {10, 11, 12}
  };
```

(c)

Outline

- Introduction
- Declaring Two-Dimensional Arrays
- Processing Two-Dimensional Arrays
- Passing Two-Dimensional Arrays to Functions
- Problem: Grading a Multiple-Choice Test
- Multidimensional Arrays

Initializing Arrays with Random Values

- Nested **for** loops are often used to process a two-dimensional array.
- The following loop initializes the array with random values between **0** and **99**:

```
for (int row = 0; row < rowSize; row++)
  {
    for (int column = 0; column < columnSize; column++)
    {
      matrix[row][column] = rand() % 100;
    }
  }
```

Printing Arrays

- To print a two-dimensional array, you have to print each element in the array using a loop like the following:

```
for (int row = 0; row < rowSize; row++)
{
  for (int column = 0; column < columnSize; column++)
    {
       cout << matrix[row][column] << " ";
    }
    cout << endl;
}
```

Summing All Elements

- To sum all elements of a two-dimensional array:

```
int total = 0;
for (int row = 0; row < ROW_SIZE; row++)
{
  for (int column = 0; column < COLUMN_SIZE;
column++)
  {
    total += matrix[row][column];
  }
}
```

Summing Elements by Column

- For each column, use a variable named **`total`** to store its sum. Add each element in the column to **`total`** using a loop like this:

```
  for (int column = 0; column < columnSize;
column++)
  {
    int total = 0;
    for (int row = 0; row < rowSize; row++)
      total += matrix[row][column];
    cout << "Sum for column " << column << " is “
         << total << endl;
  }
```

Which row has the largest sum?

- Use variables **maxRow** and **indexOfMaxRow** to track the largest sum and index of the row. For each row, compute its sum and update **maxRow** and **indexOfMaxRow** if the new sum is greater.

```
int maxRow = 0;
int indexOfMaxRow = 0;
// Get sum of the first row in maxRow
for (int column = 0; column < COLUMN_SIZE; column++)
    maxRow += matrix[0][column];
for (int row = 1; row < ROW_SIZE; row++)
{
    int totalOfThisRow = 0;
    for (int column = 0; column < COLUMN_SIZE; column++)
        totalOfThisRow += matrix[row][column];
    if (totalOfThisRow > maxRow)
    {
        maxRow = totalOfThisRow;
        indexOfMaxRow = row;
    }
}
cout << "Row " << indexOfMaxRow
    << " has the maximum sum of " << maxRow << endl;
```

Outline

- Introduction
- Declaring Two-Dimensional Arrays
- Processing Two-Dimensional Arrays
- Passing Two-Dimensional Arrays to Functions
- Problem: Grading a Multiple-Choice Test
- Multidimensional Arrays

Passing Two-Dimensional Arrays to Functions

- You can pass a two-dimensional array to a function.
- The column size to be specified in the function declaration.
- A program that for a function that returns the sum of all the elements in a matrix.

PassTwoDimensionalArray Run

PassTwoDimensionalArray.cpp 1/2

```
#include <iostream>
using namespace std;

const int COLUMN_SIZE = 4;

int sum(const int a[][COLUMN_SIZE], int rowSize)
{
    int total = 0;
    for (int row = 0; row < rowSize; row++)
    {
        for (int column = 0; column < COLUMN_SIZE; column+
+)
        {
            total += a[row][column];
        }
    }

    return total;
}
```

PassTwoDimensionalArray.cpp 2/2

```
int main()
{
    const int ROW_SIZE = 3;
    int m[ROW_SIZE][COLUMN_SIZE];

    cout << "Enter " << ROW_SIZE << " rows and "
        << COLUMN_SIZE << " columns: " << endl;
    for (int i = 0; i < ROW_SIZE; i++)
        for (int j = 0; j < COLUMN_SIZE; j++)
            cin >> m[i][j];

    cout << "\nSum of all elements is " << sum(m,
ROW_SIZE)
         << endl;

    return 0;
}
```

Outline

- Introduction
- Declaring Two-Dimensional Arrays
- Processing Two-Dimensional Arrays
- Passing Two-Dimensional Arrays to Functions
- Problem: Grading a Multiple-Choice Test
- Multidimensional Arrays

Problem: Grading Multiple-Choice Test

Key to the Questions:

	0	1	2	3	4	5	6	7	8	9
key	D	B	D	C	C	D	A	E	A	D

Students' Answers to the Questions:

	0	1	2	3	4	5	6	7	8	9
Student 0	A	B	A	C	C	D	E	E	A	D
Student 1	D	B	A	B	C	A	E	E	A	D
Student 2	E	D	D	A	C	B	E	E	A	D
Student 3	C	B	A	E	D	C	E	E	A	D
Student 4	A	B	D	C	C	D	E	E	A	D
Student 5	B	B	E	C	C	D	E	E	A	D
Student 6	B	B	A	C	C	D	E	E	A	D
Student 7	E	B	E	C	C	D	E	E	A	D

GradeExam

Run

GradeExam.cpp 1/2

```
#include <iostream>
using namespace std;

int main()
{
    const int NUMBER_OF_STUDENTS = 8;
    const int NUMBER_OF_QUESTIONS = 10;

    // Students' answers to the questions
    char answers[NUMBER_OF_STUDENTS][NUMBER_OF_QUESTIONS]
=
    {
      {'A', 'B', 'A', 'C', 'C', 'D', 'E', 'E', 'A', 'D'},
      {'D', 'B', 'A', 'B', 'C', 'A', 'E', 'E', 'A', 'D'},
      {'E', 'D', 'D', 'A', 'C', 'B', 'E', 'E', 'A', 'D'},
      {'C', 'B', 'A', 'E', 'D', 'C', 'E', 'E', 'A', 'D'},
      {'A', 'B', 'D', 'C', 'C', 'D', 'E', 'E', 'A', 'D'},
      {'B', 'B', 'E', 'C', 'C', 'D', 'E', 'E', 'A', 'D'},
      {'B', 'B', 'A', 'C', 'C', 'D', 'E', 'E', 'A', 'D'},
      {'E', 'B', 'E', 'C', 'C', 'D', 'E', 'E', 'A', 'D'}
    };
```

GradeExam.cpp 2/2

```
    // Key to the questions
    char keys[] = { 'D', 'B', 'D', 'C', 'C', 'D', 'A', 'E', 'A',
'D' };

    // Grade all answers
    for (int i = 0; i < NUMBER_OF_STUDENTS; i++)
    {
        // Grade one student
        int correctCount = 0;
        for (int j = 0; j < NUMBER_OF_QUESTIONS; j++)
        {
            if (answers[i][j] == keys[j])
                correctCount++;
        }

        cout << "Student " << i << "'s correct count is " <<
            correctCount << endl;
    }

    return 0;
}
```

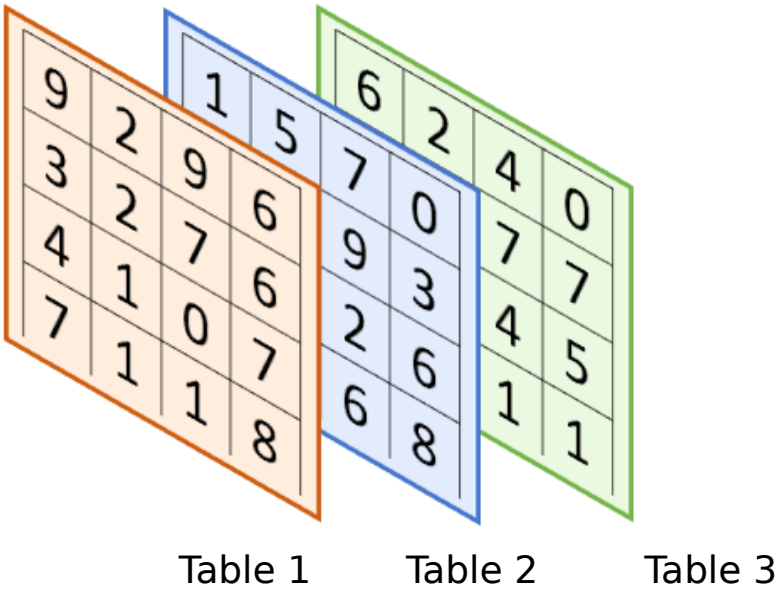
```
Student 0's correct count is 7
Student 1's correct count is 6
Student 2's correct count is 5
Student 3's correct count is 4
Student 4's correct count is 8
Student 5's correct count is 7
Student 6's correct count is 7
Student 7's correct count is 7
```

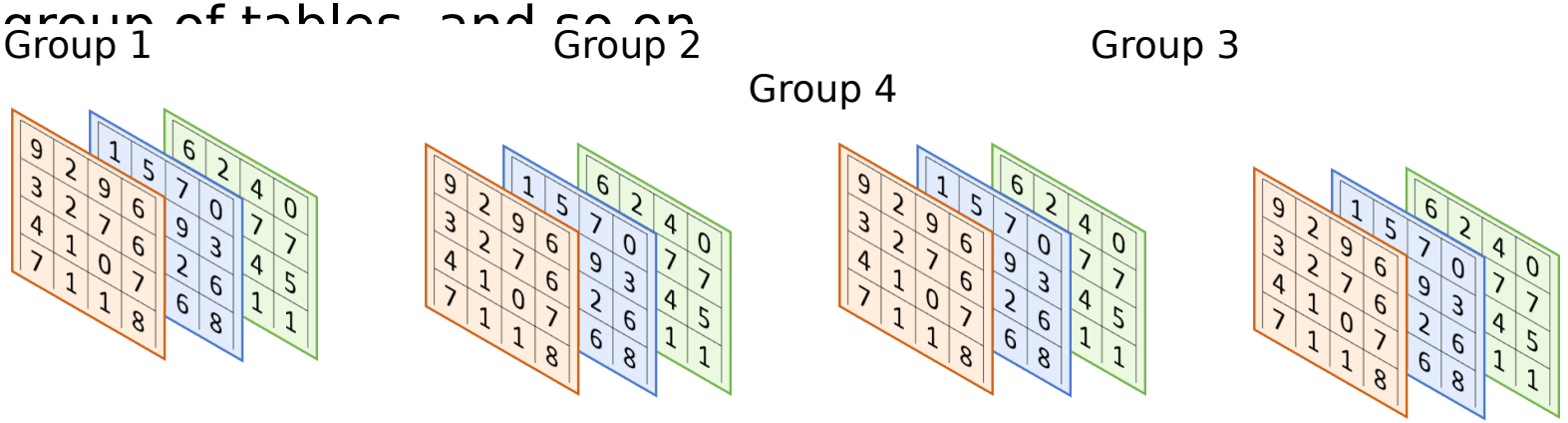
Outline

- Introduction
- Declaring Two-Dimensional Arrays
- Processing Two-Dimensional Arrays
- Passing Two-Dimensional Arrays to Functions
- Problem: Grading a Multiple-Choice Test
- Multidimensional Arrays

Multidimensional Arrays

- A *1D* array is simply one row of sequential data.
- A *2D* array is a group of rows on top of one another forming a table of rows and columns.
- A *3D* array is a group of tables.
- A *4D* array is a group of a group of tables, and so on



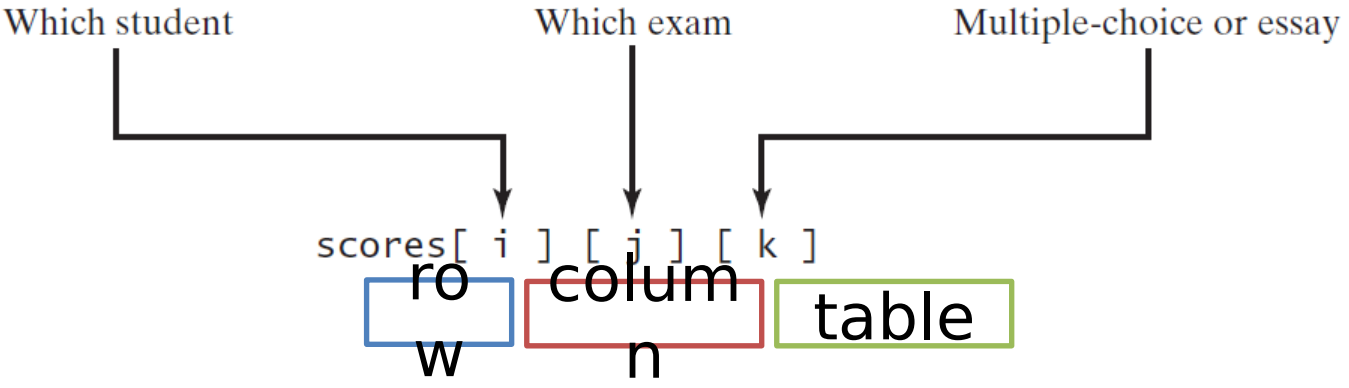


Multidimensional Arrays

You can create *n*-dimensional arrays for any integer *n*.

For example, you may use a three-dimensional array to store exam scores for a class of 6 students with 5 exams and each exam has 2 parts (multiple-choice and essay).

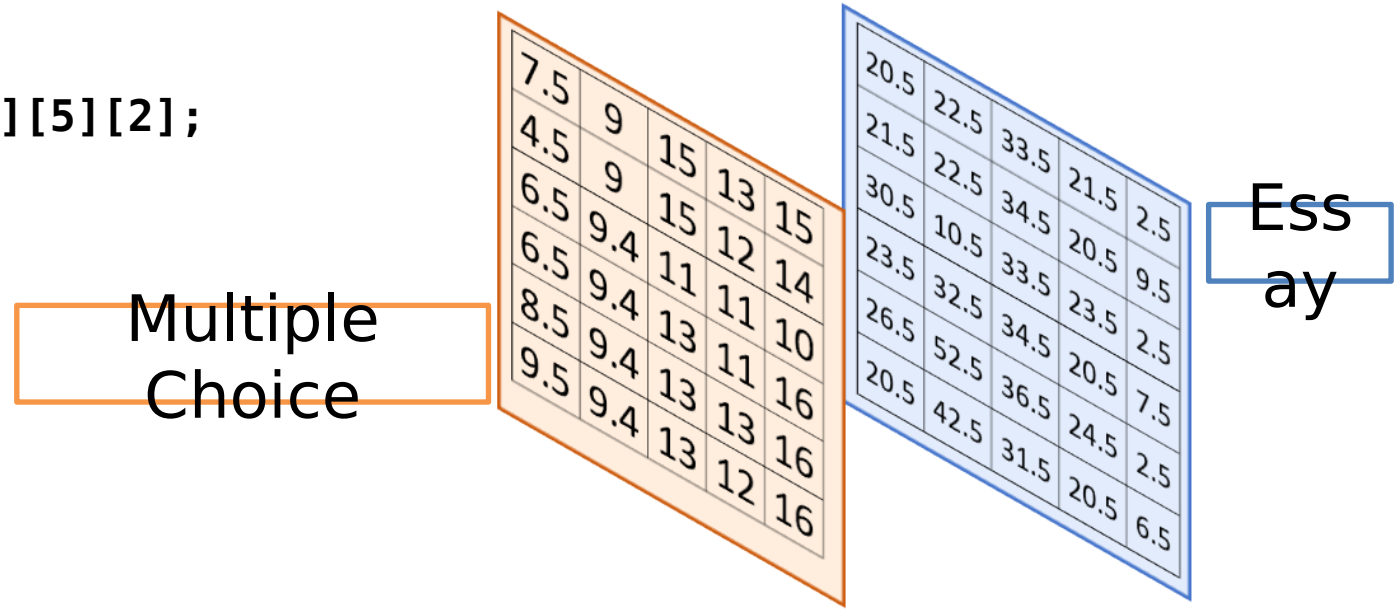
```
double scores[6][5][2];
```



Multidimensional Arrays

You can create *n*-dimensional arrays for any integer *n*.

```
double scores[6][5][2];
```



With initialization:

```
double scores[6][5][2] = {
{{7.5, 20.5}, {9.0, 22.5}, {15, 33.5}, {13, 21.5}, {15, 2.5}},
{{4.5, 21.5}, {9.0, 22.5}, {15, 34.5}, {12, 20.5}, {14, 9.5}},
{{6.5, 30.5}, {9.4, 10.5}, {11, 33.5}, {11, 23.5}, {10, 2.5}},
{{6.5, 23.5}, {9.4, 32.5}, {13, 34.5}, {11, 20.5}, {16, 7.5}},
{{8.5, 26.5}, {9.4, 52.5}, {13, 36.5}, {13, 24.5}, {16, 2.5}},
{{9.5, 20.5}, {9.4, 42.5}, {13, 31.5}, {12, 20.5}, {16, 6.5}} };
```

Problem: Daily Temperature and Humidity

- Suppose a meteorology station records the temperature and humidity at each hour of every day and stores the data for the past ten days in a text file named weather.txt.
- Each line of the file consists of four numbers that indicates the day, hour, temperature, and humidity.

```
1   1   76.4   0.92
1   2   77.7   0.93
...
10   23   97.7   0.71
10   24   98.7   0.74
```

```
Weather.exe < Weather.txt
```

A program that calculates the average daily temperature and humidity for the 10 days.

Weather Run

Weather.cpp 1/2

```
#include <iostream>
using namespace std;

int main()
{
    const int NUMBER_OF_DAYS = 10;
    const int NUMBER_OF_HOURS = 24;
    double data[NUMBER_OF_DAYS][NUMBER_OF_HOURS][2];

    // Read input using input redirection from a file
    int day, hour;
    double temperature, humidity;
    for (int k = 0; k < NUMBER_OF_DAYS * NUMBER_OF_HOURS;
k++)
    {
        cin >> day >> hour >> temperature >> humidity;
        data[day - 1][hour - 1][0] = temperature;
        data[day - 1][hour - 1][1] = humidity;
    }
```

Weather.cpp 2/2

```
    // Find the average daily temperature and humidity
    for (int i = 0; i < NUMBER_OF_DAYS; i++)
    {
        double dailyTemperatureTotal = 0, dailyHumidityTotal
= 0;
        for (int j = 0; j < NUMBER_OF_HOURS; j++)
        {
            dailyTemperatureTotal += data[i][j][0];
            dailyHumidityTotal += data[i][j][1];
        }

        // Display result
        cout << "Day " << i << "'s average temperature is "
            << dailyTemperatureTotal / NUMBER_OF_HOURS <<
endl;
        cout << "Da                                  y is "
            << dail                                 S << endl;
    }

    return 0;
```

```
Day 0's average temperature is 77.7708
Day 0's average humidity is 0.929583
Day 1's average temperature is 77.3125
Day 1's average humidity is 0.929583
...
```

Outline

- Introduction
- Declaring Two-Dimensional Arrays
- Processing Two-Dimensional Arrays
- Passing Two-Dimensional Arrays to Functions
- Problem: Grading a Multiple-Choice Test
- Multidimensional Arrays