

Name:
Sec:

09/20/2005

Lab 2.1 Coding with Parallel Arrays

Parallel arrays are two or more arrays whose corresponding components hold related information. If you need to keep track of multiple items that are related, you can create more than one array where the information is related by the placement within the arrays.

Objectives

In this lab, you process parallel arrays.

After completing this lab, you will be able to:

- ▶ Process parallel arrays.

Coding with Parallel Arrays

In the following exercises, select the correct answer for each question. Then design and write a program that uses parallel arrays.

Question	Answer (Circle the correct answer)
1. Parallel arrays can be different data types.	T F
2. Parallel arrays can be different sizes.	T F
3a. Create the design for a program that keeps track of the hits, walks, and outs of a baseball team. Use parallel arrays to keep track of each player's statistics. The player number is the index of the array. You will be reading in data from the file called PlayBall.txt , which contains data that includes the player number, hits, walks and outs. The data file will contain multiple games. Not every player will have stats in each game. The arrays should be initialized to zero (0) and should be accumulators to allow for stats for each game of the season.	
Write your design in the following space. Your design should be a list of C++ comments without any code:	

- 3b. Write a C++ program based on the design you created in Exercise 3a. You are given a start on the program which opens up the **PlayBall.txt** file and reads in the data for you. The file, named **PlayBall.cpp**, may be found on the class website.

Lab 2.2 Manipulating Data in a Two-Dimensional Array

Sometimes data is provided in a table form called a two-dimensional array, a collection of a fixed number of components arranged in rows and columns, in which all components are of the same type. To access the components of a two-dimensional array, you need a pair of indices—one for the row position and one for the column position. However, two-dimensional arrays are stored in row order in memory.

You can initialize two-dimensional arrays at declaration. Use braces for the initialization values. You can use additional braces separated by commas to subdivide each row for initialization.

Objectives

In this lab, you become acquainted with the two-dimensional array, including processing by row and column, initializing at declaration, printing, inputting, summing by column, and finding the largest element in a column.

After completing this lab, you will be able to:

- ▶ Search the first column of a two-dimensional array for a particular value.
- ▶ Use the second column of a two-dimensional array to hold data that is related to the first column.
- ▶ Initialize a two-dimensional array to zero (0) at declaration.
- ▶ Process a two-dimensional array by row.
- ▶ Process column elements of a two-dimensional array.
- ▶ Print all elements of a two-dimensional array.
- ▶ Pass a two-dimensional array to a function.
- ▶ Input values into a column of a two-dimensional array.
- ▶ Sum the values of a column of a two-dimensional array.
- ▶ Find the largest element in a column of a two-dimensional array.

Manipulating Data in a Two-Dimensional Array

In the following exercises, select the correct answer for each question. Then redesign a program to use two-dimensional arrays.

Question	Answer
1. In a two-dimensional array declaration, the first index indicates the number of rows in the table, and the second the number of columns.	T F
2. Given <code>int stats[25][3];</code> How many components are in the array stats?	

<p>3. Given:</p> <pre>int stats[2][3] = {4, 4, 4, 3, 12, 15}; int totalHits = 0; for (int row=0; row<2; row++) totalHits += stats[row][0];</pre> <p>What is the value of totalHits?</p>	
<p>4. Given</p> <pre>int stats[2][3] = {4, 4, 4, 3, 12, 15}; int atBats[2] = {0}; for (int row=0; row<2; row++) { for (int column = 0; column < 3; column++) atBats[row] += stats[row][column]; cout << "Player " << row + 1 << "was up to bat " << atBats[row] << "times." << endl; }</pre> <p>What is the output?</p>	
<p>5a. Redesign your PlayBall.cpp to use a two-dimensional array instead of parallel arrays. Add an additional column that keeps track of the number of times a player was up to bat. When printing the report, add a totals line that totals the number of hits, walks, and outs for the team. Find the player with the greatest number of hits, the player with the greatest number of walks, and the player with the greatest number of outs.</p> <p>Modify your PlayBall.cpp program to use two-dimensional arrays and name it PlayBall2.cpp.</p> <p>Enter, compile, link and execute PlayBall2.cpp.</p> <p>The following is a copy of the screen results that might appear after running your program, depending on the data entered.</p>	

This program keeps track of the hits, walks, outs, at bats, and totals of a baseball team.

Player	Hits	walks	Outs	At Bats
1	4	4	4	12
2	3	12	15	30
3	6	8	10	24
4	5	5	8	18
5	2	2	2	6
6	6	0	0	6
7	0	0	6	6
8	6	10	2	18
9	6	4	2	12
10	6	6	6	18
11	14	2	2	18
12	0	0	0	0
13	8	1	3	12
14	2	4	6	12
15	0	0	0	0
16	0	0	0	0
17	4	2	6	12
18	9	8	1	18
19	3	2	1	6
20	0	10	14	24

Totals:	84	80	88	252

Player 11 had the most hits
Player 2 had the most walks
Player 2 had the most outs
Player 2 had the most at bats
Press any key to continue

Lab 2 Finished

You have now completed Lab 2. Turn in the previous pages to the instructor and make sure you have uploaded your **PlayBall.cpp** and **PlayBall2.cpp** programs to your Educator account in your Lab-2 subfolder. Ask the instructor to examine your lab, which will be given either an acceptable or unacceptable evaluation. If unacceptable, you will be told of the problem areas and you may fix them and resubmit.

Attached at the end of this lab is a description of your second programming assignment, which will be due next Friday Sept 30. After completing this lab, now is a good time to begin thinking about how you will implement the second programming assignment.