

CSE 1301 PRACTICAL SESSION 7

Arrays (17 marks)

The aim of this practical session is to give you practice writing code that involves storing data in your program in arrays.

Coding style and documentation: You are expected to document your programs and to use a standard, clear, and consistent coding style. Up to 2 of the marks for each practical class may be deducted for poor coding style and/or inadequate documentation.

Preparation (to be completed before class) **(2 marks)**

Write all the algorithms required for the questions below, and attempt the code for all the questions.

PART 1: Arrays **(5 marks)**

- (a) Write a C function called `InputArray` that takes two parameters (i) an array of characters and (ii) an integer representing the length of the array, which reads in the elements of an array of characters.
- (b) Write a C function called `OutputArray` that takes the same parameters, and prints the elements of a character array separated by blanks.
- (c) Write an algorithm that reverses an array of characters. For instance, given the input: `D e m o n s`, which was stored in an array by the function `InputArray`, your algorithm should produce *another array* which contains the values: `s n o m e D`. Code a C function called `RevArray` that implements your algorithm.
- (d) Write an algorithm that reverses an array of characters *in place*. That is, you are *not* allowed to copy the original array into another structure. Implement your algorithm in a C function called `RevInPlace`.

Hint: look up the swap function from the lectures.

- (e) Write a program that enables you to test these functions by calling them in an appropriate order. After each of the above steps, your program should print:
 - o The action done in this step.
 - o The original array.
 - o The reversed array.

Save your program as **Prac7-1.c**.

PART 2: Finding in an array **(4 marks)**

Write a function that, given three parameters (i) an array of characters, (ii) an integer representing the length of the array, and (iii) a single character, checks to see if that character is already present in the array. Your function, `isPresent`, should return TRUE if the character is already present, and FALSE if it is not.

Write a program to test your function (you may wish to re-use the `InputArray` and `OutputArray` functions you wrote for Part 1).

PART 3: 2-D Arrays**(6 marks)**

Write an algorithm that takes as input from the user two 2-D arrays (one a 2 by 3 array, the other a 3 by 2 array) and multiplies them together, storing the result in a third array.

A reminder as to how to multiply two matrices:

- The number of rows in the first must equal the number of columns in the second and vice versa. So you multiply a 2 by 3 matrix by a 3 by 2 matrix, getting a 2 by 2 matrix. For example:

$$\begin{pmatrix} 3 & 6 & 1 \\ 2 & 1 & 4 \end{pmatrix} \begin{pmatrix} 4 & 1 \\ 2 & 5 \\ 8 & 7 \end{pmatrix} = \begin{pmatrix} 3 \times 4 + 6 \times 2 + 1 \times 8 & 3 \times 1 + 6 \times 5 + 1 \times 7 \\ 2 \times 4 + 1 \times 2 + 4 \times 8 & 2 \times 1 + 1 \times 5 + 4 \times 7 \end{pmatrix} = \begin{pmatrix} 32 & 40 \\ 42 & 35 \end{pmatrix}$$

The formula that is used can be given as

$$\begin{pmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \end{pmatrix} \begin{pmatrix} b_{11} & b_{12} \\ b_{21} & b_{22} \\ b_{31} & b_{32} \end{pmatrix} = \begin{pmatrix} \sum_{i=1}^3 a_{1i} \times b_{i1} & \sum_{i=1}^3 a_{1i} \times b_{i2} \\ \sum_{i=1}^3 a_{2i} \times b_{i1} & \sum_{i=1}^3 a_{2i} \times b_{i2} \end{pmatrix}$$

Code a C program that implements your algorithm. You will need a routine to read in a 2-D array and one to print out a 2-D array.

Note: The dimensions of the arrays can be hard-coded in your program using `#define`.

Submission:

Preparation: Show written preparation work to demonstrator at start of class for marking. If not done before class, it will not receive any marks, however you should still show it to your demonstrator during class to check your understanding.

PART 1, 2, 3: To be marked by the demonstrator during this class. No late submission.

Additional Component**(2 BONUS MARKS)**

Suppose you have a triangle defined by its three vertices (x_1, y_1) , (x_2, y_2) and (x_3, y_3) . Recall that the determinant of a matrix is given by

$$\begin{vmatrix} x_1 & y_1 \\ x_2 & y_2 \end{vmatrix} = x_1 y_2 - x_2 y_1$$

It turns out that we can calculate the area of the triangle by summing the determinants of the matrices we get by taking adjacent points. That is

$$\frac{1}{2} \begin{vmatrix} x_1 & y_1 \\ x_2 & y_2 \end{vmatrix} + \frac{1}{2} \begin{vmatrix} x_2 & y_2 \\ x_3 & y_3 \end{vmatrix} + \frac{1}{2} \begin{vmatrix} x_3 & y_3 \\ x_1 & y_1 \end{vmatrix}$$

(In fact, this generalizes to any polygon.) Write an algorithm that based on this, using arrays, calculates the area of a triangle given its co-ordinates. Implement your algorithm in a C program.