

CSE1301 Exercise Sheet 8 Structs

Exercise 1

(a) Design a struct to represent a game or competition in your favourite sport. It should contain the following sorts of information: names of each team/competitor; the kind of game (practice; pre-season competition; main season, ordinary game; or main season, final); the date; the starting time; venue; and ticket price.

(b) Declare some variables of this type. If `nextGame` and `grandFinal` are two such structs, show how you would set each member of `nextGame` to be the same as the corresponding member of `grandFinal` (if other names would be more appropriate for your sport, you may use those).

Exercise 2

Create a program to maintain a simple database for a used car lot. The used car lot sells a number of car makes and models for which they would like to maintain an inventory.

(a) Design and code an appropriate struct for storing information about each used car.

(b) Declare an array for storing a suitable number of used cars.

(c) Design and code a function to accept data for a new used car from the user. Your function should test the data entered and should return a struct storing that information.

(d) Design and code a function to store a struct in the next available element of the array. How will you keep track of the next available element? What will your algorithm do when the array is completely full?

Exercise 3

Refer to Exercise Sheet 4, exercise 3, where you wrote functions for manipulating points, but did not use structs to represent those points.

(a) Define a struct type to represent points, and rewrite your functions from Exercise 4.3(b) to work with these structs.

(b) Write a function to calculate the distance between two points.

(c) Define a structure to represent a circle.

(d) Write functions to determine

- i. the circumference of a circle,
- ii. the area of a circle,
- iii. whether or not two circles overlap,
- iv. whether or not one circle entirely contains another.

Exercise 4 (Additional Exercise)

Do one or both of the following:

(a) Write a set of C functions to manipulate complex numbers. You should end up with a library of functions that perform basic complex arithmetic and complex versions of as many of the mathematical functions in `<math.h>` as you have time for.

(b) Quaternions are a generalisation of complex numbers with four components, instead of the two (real and imaginary) that complex numbers have. Find out more about them and write a small library of functions to do arithmetic with them. Apart from textbooks, good sources for this type of mathematical information are the web sites www.mathworld.com and (sometimes) www.planetmath.org