Structures in C

The development of modern programming languages owes much to the original C language and its use of a storage type called a **structure**. In fact the idea of Object Oriented Programming (OOP) owes much to this idea as developed in C.

We have already seen a variety of ways to organize store and pass data around in our programs. One challenge that still remains is how to handle information that combines a variety of information that has different types that need to be connected to each other. For example, information on you as a student will have strings for your name, your courses, your address, and floats for your course marks, integers for things like phone numbers, street numbers etc. How can all of these various things be combined in a way that groups them together for you as a single student? One answer obviously is through the use of a structure data type.

An example of the use of a structure is shown below. I have created a sample program for tracking basic information about the Lasers hockey players. Copy the code into Code::Blocks and examine it to see how it works. Once you have taken a look, run it to see how it is handling the entry and control of a very large amount of data with what is essentially a single variable declaration; the **team lasers[25]** .

```
// A demo of the use of structures
// For good measure it includes functions to enter and display the data
// and uses a pointer to a struct which allows a local declaration of
// the structure variable.

typedef struct                                          //Defining a structure as a data type to store info
        {
        char    fName[20],
                lName[20];
        int             gamesPlayed,
                        pim,
                        goals,
                        assists,
                        number;
        }team;

#include <stdio.h>

//Function Prototypes                                   Do these at the top and put functions at the end of main
void enterData(int, team *);
void displayData(int, team *);

int main( void ) {

        int             numPlayers;
        team            lasers[25];                     //Allows us to enter up to 25 players on the team

        printf("How many players are there on your team ==>");
        scanf("%i",&numPlayers);                        //Enter the number of players... not error trapped

        enterData(numPlayers,lasers);                   //call the function, passing struct and num players

        displayData(numPlayers,lasers);                 //call the function to display the info

        return 0;
}
```

```
/*************************************************************
Function:		enterData
Receives:		the number of players (int) and the structure
Returns:		nothing
Uses:			nothing
Does:			enters team information into a structure
*************************************************************/
void enterData(int numPlay, team *players)
{
	int i;
	for(i = 0; i<numPlay;i++)                                                    //Loop to enter player data
	{
		printf("\nEnter the first name of player %i ==>",i+1);
		scanf("%s",players[i].fName);
		printf("\n\nEnter the last name of player %i ==>",i+1);
		scanf("%s",players[i].lName);
		printf("\n\nEnter the games played for player %i ==>",i+1);
		scanf("%i",&players[i].gamesPlayed);
		printf("\n\nEnter the penalty minutes for player %i ==>",i+1);
		scanf("%i",&players[i].pim);
		printf("\n\nEnter the number of goals for player %i ==>",i+1);
		scanf("%i",&players[i].goals);
		printf("\n\nEnter the number of assists for player %i ==>",i+1);
		scanf("%is",&players[i].assists);
		printf("\n\nWhat is %s %s's number  ==>",players[i].fName,players[i].lName);
		scanf("%i",&players[i].number);
		}
}


/*************************************************************
Function:		displayData
Receives:		the number of players (int) and the structure
Returns:		nothing
Uses:			nothing
Does:			outputs team information to the console
*************************************************************/
void displayData(int numPlay, team *players)
{
	int i;

	printf("\n\nHere is the data that you entered..\n\n");
	printf("Name\t\t\tNumber\tGames \t Goals\t Assists \t Penalty Minutes\n");
	printf("=============================================================\n\n");

	for(i = 0; i<numPlay;i++)                                                //Loop to display player data
	{
		printf("%10s %-10s %5i %10i %10i %10i %10i\n", players[i].fName, players[i].lName,
			players[i].number,players[i].gamesPlayed,players[i].goals,players[i].assists,players[i].pim);
	}

}
```

Assignment

Your abilities are growing and so shall your tasks, as each one should be able to grow on all of your previous learning. As with all of the previous exercises, it should be understood that each of the following should be in a "properly structured C program".

1. Create a text file called bbPlayers.txt which contains the following information:
   Line one: the number of players and the team name
   All the remaining lines will contain the following information separated by spaces:
   FirstName LastName Number Position NumGamesPlayed NumPoints NumFouls
   Have your program open this file and read in the number of players and team name and then input all the information into a structure. Display all of the input data in a clear and simple manner.

2. Adapt the program in 1) to allow the information about NumGamesPlayed, NumPoints, and NumFouls to be updated for each player. Note that simple error trapping should be done, for example, to ensure that the values can only be increased for each player. Updated information should be written back to the file.

3. Create a text file called myMusic.txt which contains the following information:
   Line One: an integer for the number of artists followed by your name
   The remaining lines will have three lines for each artist, set up as
   ArtistName
   NumberSongs
   FavTrack
   Have your program open this file and read in all of the information into an appropriate structure array. You can assume that the number of artists will be 50 or less. Your program will then display the information in a neat columnar form, with a line number beside each artist. If more than 20 artists are in the file, have them displayed in groups of 20, advanced based on user input.

4. Adapt the program above to allow:
   (i) Removal of an artist's information.
   (ii) Addition of a new artist (as long as there are less than 50)
   (iii) Updating of an existing artist, such as NumberSongs or FavTrack.
   The updated information should be able to be written back to the file at any time.