**Name: Soham Thaker SID: 011-748-159 Email: sdthaker1@myseneca.ca**

**What are structures?**

Structures are a type of a data structure that hold multiple variables of different data types within them. In order to access the variables written inside a structure, programmers need to create an object and refer to those variables by using a dot operator or an arrow notation, like, “object.variableName” or “object->variableName”.

To explain what does the definition of structure include, let us say if we define a structure called “struct Employee” with some useful data inside the structure and then create an object of that structure, “struct Employee emp”. Here, “Employee” is a user-defined data type (instead of primitive data type like, int, float, double) where a user by themselves create a data type instead of using the primitive data type. And, “emp” is an identifier of the data type “Employee” just like programmers usually write “int low;” where low is the identifier of the data type “int”.

**What have I learnt in this workshop?**

First, I learnt how to define and create a structure. Structures are usually created before the main function so that, a programmer can access the variables defined inside the main and any other functions within that file. However, this not what most programmers do. They do usually write the structure inside the header file and include that file inside any source file that uses those structure variables. This helps them with the modularity of the program. This is explained in detail in the last question of this document, “Where should a structure be declared and its advantages?”

Second, I learnt how to create an object of a structure. The object itself can also be referred to as an identifier of a structure data type. And the keyword, “struct” must be written to create an object. C language also provides a “typedef” functionality where, a programmer does not need to write the keyword “struct” every time they create an object. Also, another useful information I learned about objects is that programmers can create any number of objects of a structure. This is similar concept to the OOP languages where programmers can create, any number of objects of a class.

Third, I learnt how to write an array of a structure and what are the applications of array of structure. An array of a structure helps in categorizing the variables written inside a structure. As witnessed in workshop 5, an array of structures, can help programmers store together, data of a similar type. For example, employees in a company have a data that is common, which is, ID of an employee, age of an employee and salary of an employee, among other data. By using array of structure, programmers can refer to these employees inside a C program as Employee 1, Employee 2, Employee 3 and so on, by assigning these employees an element of an array. Since an array of the structure is created, programmers can easily bind together information like, ID number, Age and Salary of an employee. And if the programmers want to increase the size of a company by employees, they can do so by changing the size of the array of the structure. Programmers can update the information, by minimal amount of work.

Lastly, I learnt the applications of a switch case within a program. Switch cases work similarly as if-else statements. Usually, in a switch case, a number or an alphabet is passed and if argument matches with the any of the cases, it will execute the code written inside the matched case and will break the case after the code is executed. If none of the cases match, then control goes to the default case, executes the code within the default case and breaks the case. Fundamentally if-else statements and switch cases are similar, however different in syntax.

**Advantages of using array of structures as compared to using parallel arrays.**

Parallel arrays are arrays which hold similar data at their respective indexes. By using a for loop, programmers can print or change the data held into these arrays. However, parallel arrays have their own limitations when using them to deal with related data.

Using array of structures is more feasible when dealing with related data because structures allow flexibility of writing multiple data types within a structure and when an array of structure is created, that specific array can store, change, update or print the variables within the structure. The main benefit here is that a normal array only holds a value of one data type, whereas an array of structure holds values of multiple data types which are defined within a structure.

Also, for parallel arrays, programmers would have to individually compare the data and assign them the new values accordingly and in this way the program would get lengthier which might make it difficult to identify and assign these arrays with their respective values. Whereas, in an array of structures, a data related to certain employee can be changed by using the index of an array along with the variables inside a structure of an array.

For example, three inter-connected data of 100 employees is to be created and stored in variables namely employee ID, employee age and employee salary. Now if programmers were to write a data of 100 employee by using parallel arrays, they would have to create, 100 \* 3 = 300 arrays, and this would add to more confusion regarding the data stored in each array and also take more space on the disk. Whereas, in an array of structures, 100 arrays will be created, which hold the information of those 100 employees. Each employee is referred to as the index number of an array of structure, which makes it very easy to refer to an individual employee and data pertaining to that employee like employee ID, employee age and employee salary.

In a nutshell, an array of structure provides Clarity, Conciseness, Understandability, Efficiency, Memory allocation in a proper manner & saves disk space.

**Where should a structure be declared and its advantages?**

A structure should always be declared in a header file and include that header file into a source file if that source file needs to utilize those structure variables.

Let us say a programmer creates a C program with multiple .c files within a project and they have created a structure that contains multiple data within them. If he or she is utilizing that structure within more than one source file, the least efficient way to declare and define the structures is to write that structure definition in each and every file which is also more work for a programmer. If they want to use the same variables into another .c file within the same project, they will have to write, or copy paste the structure again and again and again.

On the other hand, if a header file called “student” is used to define and declare a structure, only change a programmer has to do is write #include “student.h” at the beginning of a .c file that would utilize the structure variables and he or she will have an automatic access to all the structure variables defined in that structure, stored in a header file. This will save a lot of the programmers’ time and hassle to rewrite the structure individually.

This method also provides programmers with a solution to modularity. Let us say they are using multiple functions in multiple .c files under one project, then they can use a header file to write the prototype of the function and include that header file into a .c file where these functions are to be used. This helps with the clarity of the code.

Usually, constants, macros, system wide global variables, function prototypes, structures among many other contents can be written inside a header file, which might be used later in the program.