Lab 2 – Structures

Lab Outcomes:

- **Understanding Arrays in C++:** Learn about arrays as collections of elements of the same type, stored in contiguous memory locations, and how they allow easy access to data through indexing.
- **Types of Arrays:** Differentiate between 1D arrays and multidimensional arrays, with a focus on 2D arrays, which are used to represent tabular data in rows and columns.
- **Introduction to Structures in C++:** Explore structures as user-defined data types in C++ that allow grouping of items with different data types under a single name, facilitating better organization and readability of code.
- **Declaring and Using Structures:** Learn the syntax for declaring structures and defining structure members. Understand how to use structure variables to store related information about multiple items.
- **Working with Structure Members:** Understand the use of the dot operator to access and manipulate structure members, including initializing and displaying values of structure members.
- **Passing Structure Variables to Functions:** Gain experience in passing structure variables by value and by reference to functions, and learn how to define function prototypes for structures.

We often come around situations where we need to store a group of data whether of similar data types or non-similar data types. We have seen Arrays in C++ which are used to store set of data of similar data types at contiguous memory locations.

What is Array?

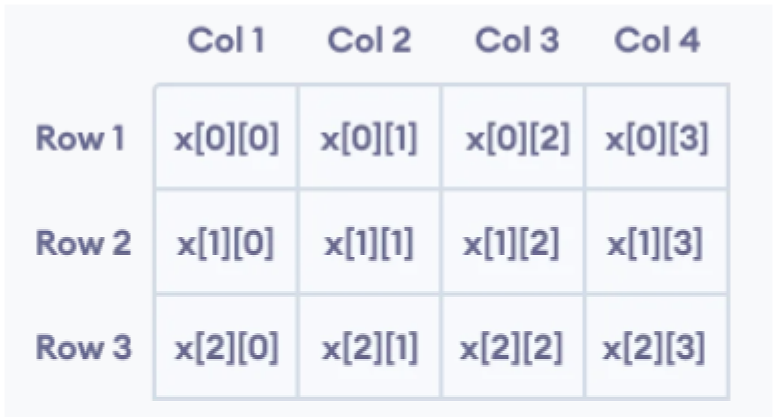
An **array** is a collection of elements of the same type placed in contiguous memory locations that can be individually referenced by using an index to a unique identifier.

Types of array:

- **1D array:**
  Syntax:

```
string cars[4];
string cars[4] = {"Volvo", "BMW", "Ford", "Mazda"};
int myNum[3] = {10, 20, 30};
```

- **Multidimensional array:** array of an array is called multidimensional array. Normally we use 2D array in C++.
  Syntax:
  Int x[3][4];



	Col 1	Col 2	Col 3	Col 4
Row 1	x[0][0]	x[0][1]	x[0][2]	x[0][3]
Row 2	x[1][0]	x[1][1]	x[1][2]	x[1][3]
Row 3	x[2][0]	x[2][1]	x[2][2]	x[2][3]

Unlike Arrays, **Structures in C++** are user defined data types which are used to store group of items of non-similar data types.

What is a structure?

A structure is a user-defined data type in C/C++. A structure creates a data type that can be used to group items of possibly different types into a single type.

For example: You want to store some information about a person: his/her name, age and salary. You can easily create different variables name, age, salary to store these information separately.

However, in the future, you would want to store information about multiple persons. Now, you'd need to create different variables for each information per person: name1, age1, salary1, name2, age2, salary2

You can easily visualize how big and messy the code would look. Also, since no relation between the variables (information) would exist, it's going to be a daunting task.

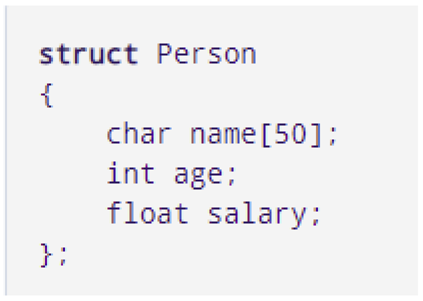
A better approach will be to have a collection of all related information under a single name Person, and use it for every person. Now, the code looks much cleaner, readable and efficient as well.

This collection of all related information under a single name **Person** is a structure.

How to declare a structure in C++ programming?

The **struct** keyword defines a structure type followed by an identifier (name of the structure).

Then inside the curly braces, you can declare one or more members (declare variables inside curly braces) of that structure. For example:



```
struct Person
{
    char name[50];
    int age;
    float salary;
};
```

Here a structure person is defined which has three members: name, age and salary.

Differences between simple datatype variables vs structure variables.

Int Datatype Variable	**Structure Variable**

<u>Variable Declaration</u>
 	
int var; // int variable declared
	<u>Variable Declaration</u>
	
	struct Person{
	int age;
	};
	
	void main(){
	Person var; // structure variable declared
	}

<u>Variable Initialization</u> var = 5;	**<u>Variable Initialization</u>** var.age = 5;
<u>Passing variable through function call</u> Display(var);	**<u>Passing variable through function call</u>** Display(var);

<u>Receiving argument in function prototype</u>
 	
void Display(int var); // pass by value
void Display(int &var); // pass by reference
	<u>Receiving argument in function prototype</u>
	
	void Display(Person var); // pass by value
	void Display(Person &var); // pass by reference

<u>Display function</u>	**<u>Display function</u>**

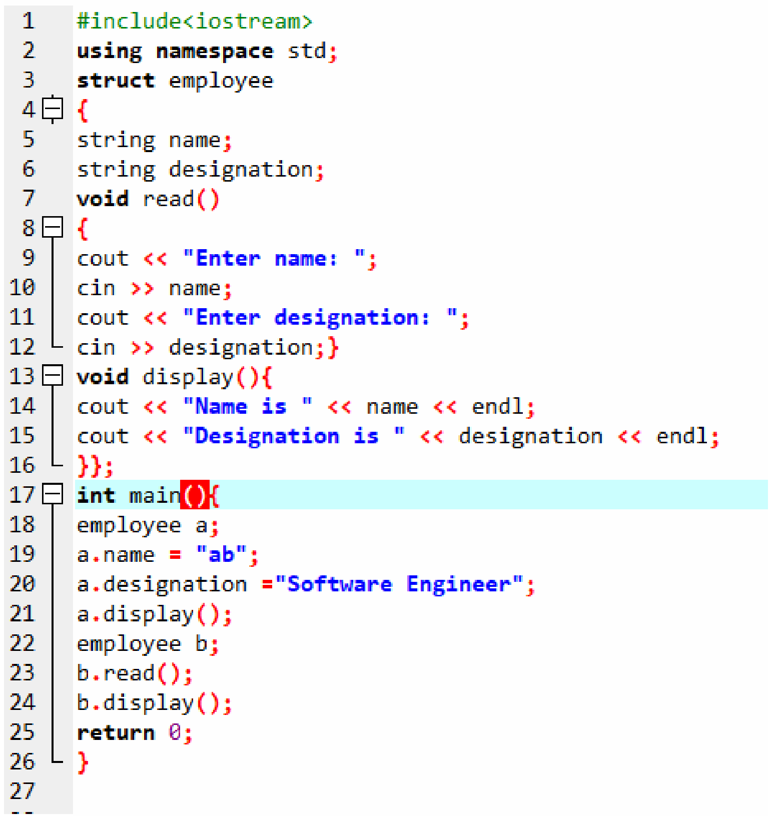
void Display(int &var)
{
  cout << var << endl;
}
	void Display(int &var)
	{
	  cout << var.age << endl; // dot operator is used to print the variables of structure.
	}

<u>**Read function**</u>
 	
void Read(int &var)
{
  cin >> var;
}
	<u>**Read function**</u>
	
	void Read(Person &var)
	{
	  cin >> var.age;
	}

Example

Implement the below tasks.

- Create a structure “Employee” and add two variables in it for storing Name and Designation.
- Initialize the variables using hardcoded values.
- Display the outputs.
- Now create **read** and **display** function for reading values and showing them on console.



```
#include<iostream>
using namespace std;
struct employee
{
string name;
string designation;
void read()
{
cout << "Enter name: ";
cin >> name;
cout << "Enter designation: ";
cin >> designation;}
void display(){
cout << "Name is " << name << endl;
cout << "Designation is " << designation << endl;
}};
int main(){
employee a;
a.name = "ab";
a.designation ="Software Engineer";
a.display();
employee b;
b.read();
b.display();
return 0;
}

```

Lab Tasks

TASK 1:

Create a structure **"STUDENT"** which have data members **name, rollNo**, **Uni_Name, Year and Semester.**

After defining the structure, this program takes the data member's values from the users for 2 students. It displays the record of these 2 students on the console.

TASK 2: Implement the below tasks.

- Create a structure "Fruit" add 3 data members to store information Fruit name, price and quantity.
- Create 5 Fruits with their names and add it price and quantity.
- Create variable Grand_total and show the total cost of fruits (for example: Apple Price is 10 rupees and orange price is 15 and you bought 10 apples and 10 oranges. Grand_Total=250)

TASK 3: Create a structure **Item** with members**:**

- Item Code
- Item Name
- Price
- Quantity

Input details of 5 items.
Calculate the total value of stock (Price × Quantity for each item, then sum).

Task 4:Create a structure course with some attributes i.e course_ID, course_title, credit_hrs . Then Implement

following 5 functions (CREATE, READ, UPDATE, DELETE,SEARCH operations):

1. addAStudent

2. updateAStudent

3. deleteAStudent

4. searchAndDisplayAStudent

5. displayAllstudents

After that, create an array of 5 courses in main function. Create a menu in main function to enable user to select and perform the operations we created above. Program must not exit until and unless user wants to do so.

Sample Menu:

Main Menu

Press 1 to add a Course

Press 2 to update a Course

Press 3 to delete a Course

Press 4 to search and display a Course

Press 5 to display all Courses

Press e to exit the program

Task 5: Create a class Android_Device. The data members of the class are IMEIno (int), Type (String), Make (String), Modelno (int), Memory(float), Operating_System(String). Then Implement member functions to:

1. Set the values of all data members.

2. Display the values of all data members

Lab # 01 Marks distribution

	ER1	**ER6**	**ER8**
Task	3 points	3 points	4 points

Lab # 01 Rubric Evaluation Guideline:

#	**Qualities & Criteria**	**0 < Poor <=0.5**	**0.5 < Satisfactory <= 1.5**	**1.5 < Excellent <=2**
ER1	**Task Completion**	Minimal or no program functionality was achieved.	Some tasks were completed, but the program has errors or incomplete functionalities.	All tasks were completed, and the program runs without errors.
#	**Qualities & Criteria**	**0 < Poor <=0.5**	**0.5 < Satisfactory <= 1.5**	**1.5 < Excellent <=2**
ER6	Program Output	Output is inaccurate or poorly presented.	Output is mostly accurate but may lack labels, captions, or formatting.	Output is clear, accurate, and well presented with labels, captions, and proper formatting.
#	**Qualities &**	**0 < Poor <= 1.5**	**1.5< Satisfactory <= 3**	**3< Excellent <= 4**

	Criteria			
ER8	Question & Answer	Answers some questions but not confidently or based on lab task knowledge.	Answers most questions confidently and based on lab task knowledge.	Answers all questions confidently and demonstrates a deep understanding of the given lab task.