SLIIT ACADEMY

Higher Diploma in Information Technology
Year 1, Semester 1

Introduction to Programming(C++)

Lecture 10 : Structures in C++

Intended Learning Outcomes

End of this lecture you will be able to learn ,

LO1 : Understand the concept of Structures in C++

LO2 : Understand the concept of Structures in C++ with Pointers and functions.

What is a Structure?

- A structure is a user-defined data type in C++.
- A structure creates a data type that can be used to group items of possibly different types into a single type.
- In other words, Structure is a collection of variables of different data types under a single name.
- A struct is **heterogeneous** in that it can be composed of data of different types.

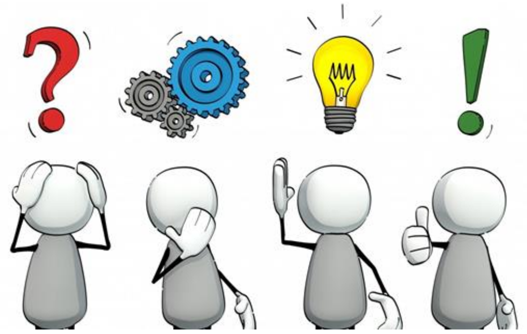
What is a Structure?

- Structures hold data that belong **together**.
- Examples:
    - Student : student id, name, major, gender, start year, …
    - Bank Account: account number, name, currency, balance, …
    - Address book: name, address, telephone number
    - Person : person name , age ,address

SLIIT ACADEMY

What is a Structure?

- Individual components of a struct type are called **members (or fields).**
- Members can be of **different types (simple, array or struct).**
- A struct is named as a whole while individual members are named using field identifiers.
- Complex data structures can be formed by defining arrays of structs.



How to declare a structure in C++ programming?

- To create a structure, use the **struct keyword** and declare each of its members inside curly braces.

```
struct <structure-name>
{
    <type> <identifier_list>;
    <type> <identifier_list>;
};
```

- When a structure is created, no memory is allocated.
- The structure definition is only the blueprint for the creating of variables.

Each identifier defines a member of the structure.

Declaring Structures

Does Not Reserve Space

```
struct my_example
{
        int label;
        char letter;
        char name[20];
} ;
```

Reserve Space

```
struct my_example
{
        int label;
        char letter;
        char name[20];
} mystruct ;
```

Structure Examples

```
struct Date {
    int day;
    int month;
    int year;
    } ;
```

The "Date" structure has 3 members, day, month & year.

```
struct StudentInfo {
    int Id;
    int age;
    int gender;
   double CGA;
    } ;
```

The "StudentInfo" structure has 4 members of different types.

```
struct BankAccount{
  char Name[15];
  int AcountNo;
  double balance;
  Date Birthday;
};
```

The "BankAcount" structure has simple, array and structure types as members

Practice Question 01

- Create the Structure called **Employee** with the following members.

```
• EmpID int
• Empname string
• Age int
• Salary float
```

```
struct Employee{
     int EmpID ;
     string Empname;
     int Age;
     float Salary;
 };
```

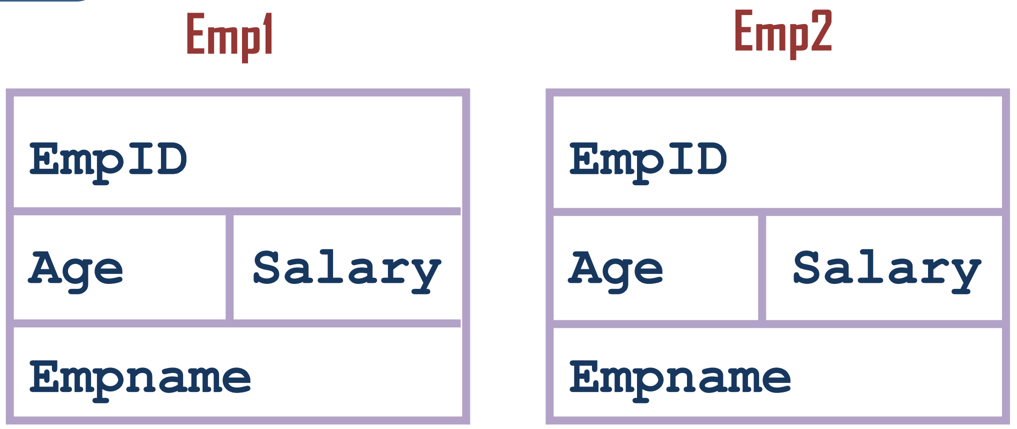
How to define a structure variable?

```
struct <struct-type> <identifier_list>;
```

- For the Employee structure create two structure variables called Emp1 and Emp2.

```
struct Employee Emp1,Emp2 ;
```

Emp1 and Emp2 are variables of Employee type

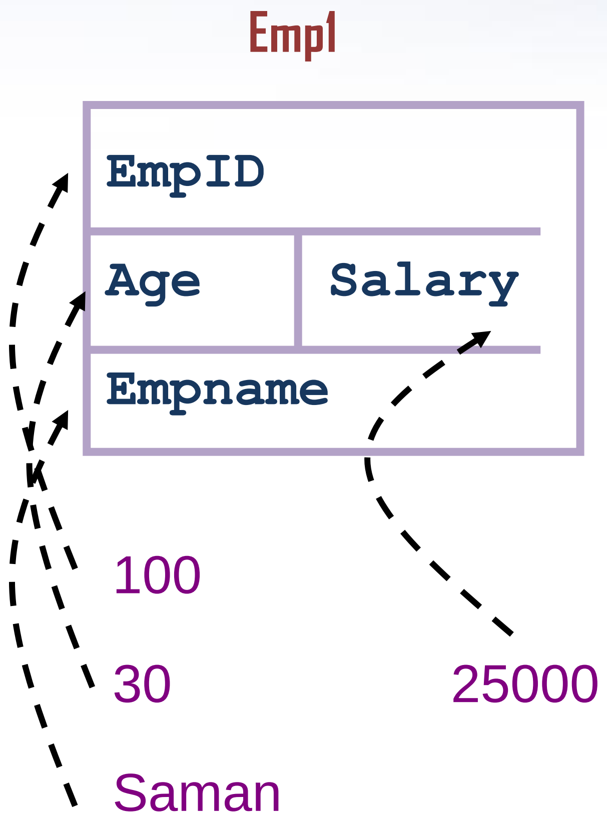


Accessing Members of a Structure

Method 01 :The members of a struct type variable are accessed with the dot (.) operator:

```
<struct-variable>.<member_name>;
```

```
Emp1.EmpID = 100 ;
Emp1.Empname = "Saman" ;
Emp1.Age = 30;
Emp1.Salary = 25000;
```



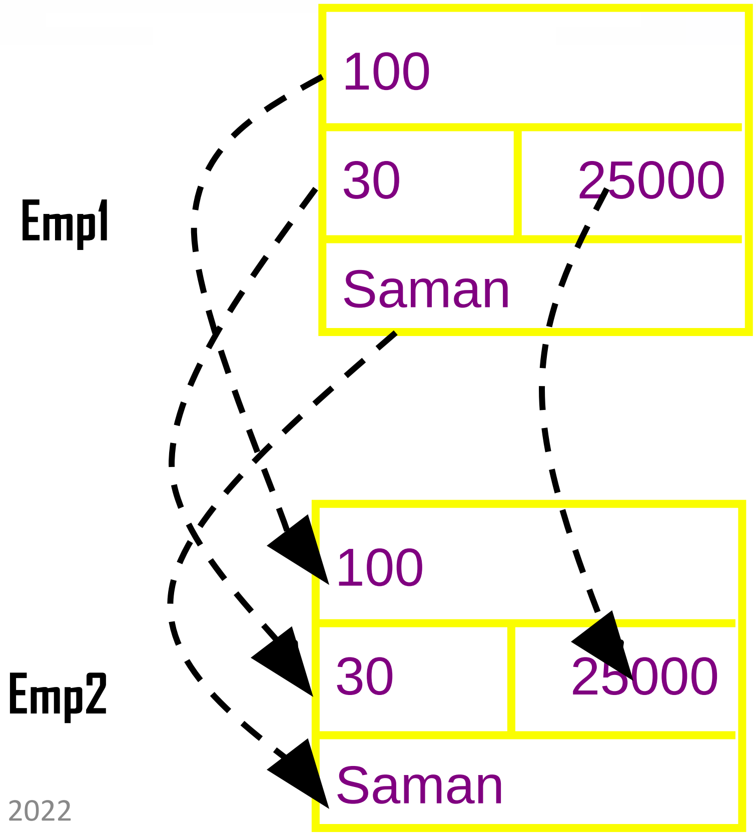
Method 02 : listing the element's value inside curly braces, with each value separated by a comma.

```
struct Employee Emp1 = {100,"Saman",30,25000}
```

Struct-to-Struct Assignment

- The values contained in one struct type variable can be assigned to another variable of the same struct type.

```
Emp2 = Emp1
```



One Structure in Multiple Variables

- You can use a comma (,) to use one structure in many variables:

```
struct {
    <type> <identifier_list>;
    <type> <identifier_list>;
} <structure-name1>, <structure-name2>;
```

One Structure in Multiple Variables

```
struct {
  string brand;
  string model;
  int year;
} myCar1, myCar2; // We can add variables by separating them with a comma here

// Put data into the first structure
myCar1.brand = "BMW";
myCar1.model = "X5";
myCar1.year = 1999;

// Put data into the second structure
myCar2.brand = "Ford";
myCar2.model = "Mustang";
myCar2.year = 1969;

// Print the structure members
cout << myCar1.brand << " " << myCar1.model << " " << myCar1.year << "\n";
cout << myCar2.brand << " " << myCar2.model << " " << myCar2.year << "\n";
```

User Defined Data Types (typedef)

- The C++ language provides a facility called *typedef* for creating synonyms for previously defined data type names.

```
typedef type newname;
```

- For example, the declaration:

```
typedef int Length;
```

makes the name Length a synonym (or alias) for the data type int.

- The data “type” name Length can now be used in declarations in exactly the same way that the data type int can be used:

```
Length   a, b, len ;
Length   numbers[10] ;
```

SLIIT ACADEMY

User Defined Data Types (typedef) & Structures

- Often, *typedef* is used in combination with *struct* to declare a synonym (or an alias) for a structure:

```
typedef struct Employee{
    int EmpID ;
    string Empname;
    int Age;
    float Salary;
 }Emp;
```

Define the Structure.

The Alias is Emp

Create the Struct variable

```
Emp e1;
```

C++ Structures and Functions

Passing structure to function in C++

A structure variable can be passed to a function in similar way as normal argument.

Returning structure from function in C++:

A structure variable can be returned from a function.

SLIIT ACADEMY

Practice Question 02

- Write a function called **getEmployeeData()** which is the data type of Employee that reads the details of Employee and store them in the variable of the Employee structure.

```
Employee getEmployeeData (Employee emp);
```

- Write a function called **printEmployeeData()** to print the Employee details.

```
void printEmployeeData(Employee emp);
```

- Call the **getEmployeeData()** and **printEmployeeData()** functions in the main function.

C++ Pointers to Structures

- A pointer variable can be created not only for native types like (int, float, double etc.) **but they can also be created for user defined types like structure.**
- Can have pointers to int, char and other data-types, also have pointers pointing to structures.
- These pointers are called **structure pointers.**

C++ Pointers to Structures

- Can have pointer to a single structure variable, but it is mostly used when dealing with array of structure variables.
- However, if we are using pointers, it is far more preferable to access struct members using the **->** operator, since the . operator has a higher precedence than the * operator.

Summary

- What is a Structure?
- How to declare the structure in C++ and define the structure variable?
- Accessing Members of a Structure
- Struct-to-Struct Assignment
- One Structure in Multiple Variables
- User Defined Data Types (typedef)
- C++ Structures and Functions
- C++ Pointers to Structures