# Structures

Prepared By: Tasmiah Tamzid Anannya, Lecturer, Cs Dept, AIUB

# The concept of structures

- Structure: a tool for grouping ***heterogeneous*** elements together.
- Array: a tool for grouping ***homogenous*** elements together
- Example: storing calendar dates (day, month, year)
- Version1: using independent variables:
- int month = 9, day = 25, year = 2004;
- Using this method, you must keep track of three separate variables for each date that you use in the program—variables that are logically related. It would be much better if you could somehow group these sets of three variables together. This is what the structure in C allows you to do !

# Structure

- structure is a collection of variables (can be of different types) under a single name.
- **For example:** You want to store information about a person: his/her name, citizenship number and salary. You can create different variables- *name*, *citno*, *salary* to store those information separately.
- What if you need to store information of more than one person? Now, you need to create different variables for each information per person: *name1*, *citno1*, *salary1*, *name2*, *citno2*, *salary2*.
- A better approach would be to have a collection of all related information under a single name ***person*** structure, and use it for every person.

# How to define a structure?

- Keyword struct is used while creating a structure.

```
struct structure_name
{
    data_type member1;
    data_type member2;
    .
    .
    data_type memeber;
};
```

- For Example,

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

# How to declare structure variables?

```
struct Person
{
    char name[50];
    int citNo;
    float salary;
}p1, p2;  //variable p1  and p2 is declared as a Person variable
```

**Another way of creating a structure variable is:**

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

int main()
{
    struct Person p1, p2; //variable p1  and p2 is declared as a Person variable

}
```

In C++, the struct keyword is optional before in declaration of a variable. In C, it is mandatory.

# How to Access members of a structure?

- There are two types of operators used for accessing members of a structure.
  - Member operator(.)
  - Structure pointer operator(->) (will be discussed in structures and pointers)
- Suppose, you want to access salary of p2.

*p2.salary*

# More on Structure

- Structure is a user defined data type.
- No memory is allocated while the structure is defined.
- Memory is created when the variables are created.

# Example: Structure

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

- Defines type struct date, with 3 ***fields*** of type int. The names of the fields are local in the context of the structure.

```
struct date today, purchaseDate;
```

- Defines 3 variables of type struct date

```
today.year = 2004;
today.month = 10;
today.day = 5;
```

# Example

```
#include <stdio.h>
struct date
{
int month;
int day;
int year;
};

int main ()
{

struct date today;
printf ("Enter today's date (mm dd yyyy): ");
scanf ("%d %d %d", &today.month, &today.day, &today.year);
printf("Today's date: %d/%d/%d", today.day, today.month, today.year);

return 0;
}
```

# Arrays of Structure

```
#include<stdio.h>
struct Employee
{
    int employee_id;
    float sal;
};
int main()
{
    struct Employee emp[5];
    int i, j;
     for(i = 0; i < 3; i++)
    {
        printf("\nEmployee ID:\t");
        scanf("%d", &emp[i].employee_id);
        printf("\nEnter Salary:\t");
        scanf("%f", &emp[i].sal);
    }
    printf("\nDisplaying Employee record:\n");
    for(i = 0; i < 3; i++)
    {
        printf("\nEmployee name is %d", emp[i].employee_id);
        printf("\nSlary is %f", emp[i].sal);
    }
}
```