Structure & Enum

SESSION 10

Structure

A data type that holds different types of data within a single group

```
struct Books {
   public string title;
   public string author;
   public string subject;
   public int book_id;
};
```

Defining a Structure

```
Start
Structure employee
{
char name[10]
char address[20]
float salary
}
End
```

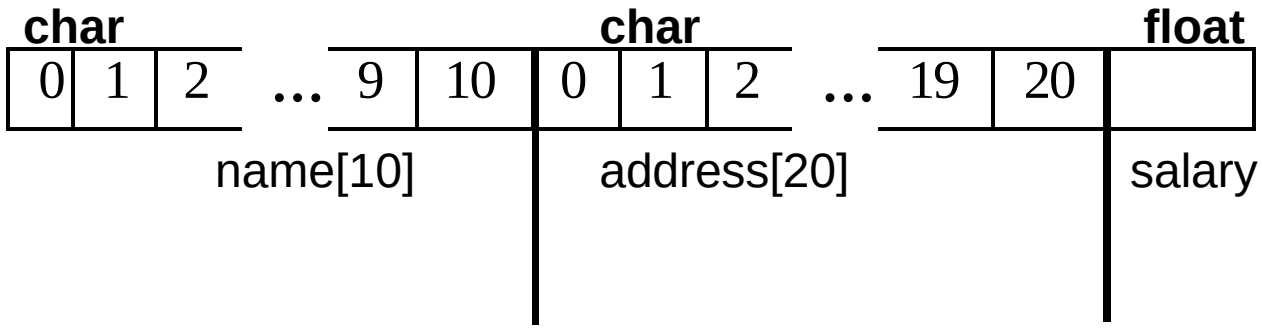
Structure keyword declares a structure in algorithms

name, address and salary are the members of the structure

employee is the structure name

Defining a Structure (Contd.)

- The members within the structure **employee** can be visualized as:



Defining a Structure (Contd.)

```
Start
Structure employee
{
char name[10]
char address[20]
float salary
}
Structure employee e1
End
```

- Structure **e1** of the type employee is created

Accessing the Members of a Structure

Members of structure are accessed as:

Structure variable.Member variable

Example: To access members of structure **e1**

e1.name

e1.address

e1.salary

Structure variables can be assigned values

e1.name = “Jackson”

e1.address = “15/2, New York”

e1.salary = 500,000

Structures (Example)

To calculate the area of a rectangle:

```
Start
Structure rectangle
{
int length
int breadth
}
Structure rectangle rect
declare area as integer
rect.length = 10
rect.breadth = 2
area = rect.length * rect.breadth
End
```

Variant of a Structure

```
Start
Structure rectangle
{
int length
int breadth
}
Structure rectangle rect = {10,2}
End
```

Variable **rect** is defined and the values 10 and 2 is assigned for its members

```
struct Books {
   public string title;
   public string author;
   public string subject;
   public int book_id;
};
public class testStructure {
   public static void Main(string[] args) {
      Books Book1;
     Books Book2;

Book1.title = "C# Programming";
      Book1.author = "Nuha Ali";
     Book1.subject = "C# Programming Tutorial";
      Book1.book_id = 6495407;
      Book2.title = "Telecom Billing";
       Book2.author = "Zara Ali";
       Book2.subject =  "Telecom Billing Tutorial";
       Book2.book_id = 6495700;

        Console.WriteLine( "Book 1 title : {0}", Book1.title);
      Console.WriteLine("Book 1 author : {0}", Book1.author);
      Console.WriteLine("Book 1 subject : {0}", Book1.subject);
     Console.WriteLine("Book 1 book_id :{0}", Book1.book_id);
      Console.WriteLine("Book 2 title : {0}", Book2.title);
      Console.WriteLine("Book 2 author : {0}", Book2.author);
      Console.WriteLine("Book 2 subject : {0}", Book2.subject);
      Console.WriteLine("Book 2 book_id : {0}",
Book2.book_id);
       Console.ReadLine();
    }
}
```

Structures

- Custom data Types
- Can have methods
- Can have constructors
- Cannot implement inheritance

```
…
struct TestStruct
{
public TestStruct()
      {
      //Constructor Implementation
      }

public Method1()
      {
      //Method1 Implementation
      {

public int dataMember;
}
…
```

```
struct Books
   {
      public string title;
      public string author;
      public string subject;
      public int book_id;
      public Books(string t, string a, string s, int id)
      {
         title = t;
         author = a;
         subject = s;
         book_id = id;
      }
      public void Display()
      {
         Console.WriteLine("Title : {0}", title);
         Console.WriteLine("Author : {0}", author);
         Console.WriteLine("Subject : {0}", subject);
         Console.WriteLine("Book_Id :{0}\n",
book_id);
      }
   };
```

```
class Program
   {
      public static void Main(string[] args)
      {
         Books Book1 = new Books("C#
Programming", "Nuha Ali", "C# Programming
Tutorial", 6495407);

         Books Book2 = new Books("Telecom Billing",
"Zara Ali", "Telecom Billing Tutorial", 6495700);

         Book1.Display();
         Book2.Display();
        Console.ReadLine();
      }
   }
```

Enumerators (1)

- They are a set of named constants.

```
using System;
public class food
{
 public enum foodType
      {
      Pizza,
      Pasta,
      Spaghetti,
      }

 public void GetFoodOrder(String CustName, foodType order)
      {
      //Process FoodOrder
      }

 static void Main()
 {
 food myDinner = new food();

 myDinner.GetFoodOrder("Scooby", food.foodType.Pizza);
 }
}
```

Enumerators (2)

- Enumerators in C# have numbers associated with the values.
- By default, the first element of enum is assigned a value of 0 and is incremented for each subsequent enum element.

Enumerators (3)

- The default value can be overridden during initialization.

```
…
public enum foodType
      {
      Pizza = 1,
      Pasta = 3,
      Spaghetti = 5,
      }
…
```