Case Study

Problem Statement CompTech is a medium-sized organization that is involved in the sale of computer hardware. It also has a small software development department, which develops customized software and Web sites for different organizations. CompTech has been till now manually maintaining the records of its employees, which is time consuming and decreases the efficiency of the HR department staff. How can CompTech solve the problem?

Solution CompTech requests one of the programmers, Harsh, who has been working for the past two years in the software development department, to create an application, which will enable the HR department to automate the task of entering employee data. In order to judiciously utilize memory space, Harsh decides to use the structure feature of C#. A structure is a value type that allows us to organize related data of different data types. After much research, Harsh creates the following C# application:

```
using System;
using System.Collections.Generic;
using System.Text;
namespace StructureExample
{
    public struct Employee
    {
        private string emp_details;

        public string Write
        {
            get { return emp_details; }
            set { emp_details = value; }
        }
        public string Read()
        {
            return Console.ReadLine();
        }
    }
    public struct EmployeeDetails
    {
        Employee name;
        Employee address;
        Employee telephone;
        Employee dept;
        Employee sal;
        public Employee Name
        {
            get { return name; }
            set { name = value; }
        }
        public Employee Address
        {
            get { return address; }
            set { address = value; }
        }
        public Employee Telephone
        {
            get { return telephone; }
            set { telephone = value; }
        }
        public Employee Department
        {
            get { return dept; }
            set { dept = value; }
        }
        public Employee Salary
```

```
        {
            get { return sal; }
            set { sal = value; }
        }
        public void ShowAllDetails()
        {
            Employee emp = new Employee();
            Console.WriteLine("Employee Details Registration System.");
            Console.WriteLine("-----------------------");
            Console.Write("Enter the Name of the Employee: ");
            name.Write = emp.Read();
            Console.Write("Enter the Address of the Employee: ");
            address.Write = emp.Read();
            Console.Write("Enter the Telephone of the Employee: ");
            telephone.Write = emp.Read();
            Console.Write("Enter the Department of the Employee: ");
            dept.Write = emp.Read();
            Console.Write("Enter the Salary of the Employee: ");
            sal.Write = emp.Read();
        }
    }
    class ShowEmployeeDetails
    {
        static void Main(string[] args)
        {
            string input = "";
            EmployeeDetails emp1 = new EmployeeDetails();
            emp1.ShowAllDetails();
            Console.WriteLine();
            Console.WriteLine("Showing Employee Details");
            Console.WriteLine("-----------------------");
            Console.WriteLine("Name: {0}", emp1.Name.Write);
            Console.WriteLine("Address: {0}", emp1.Address.Write);
            Console.WriteLine("Telephone: {0}", emp1.Telephone.Write);
            Console.WriteLine("Department: {0}", emp1.Department.Write);
            Console.WriteLine("Salary: {0}", emp1.Salary.Write);
            Console.WriteLine("-----------------------");
            Console.WriteLine("Press y to save the record....");
            input = Console.ReadLine();
            if (input == "y")
            {
                Console.WriteLine("Record Saved.");
            }
            else
            {
                Console.WriteLine("Record not saved.");
            }
        }
    }
}
```

Remarks

The use of structure in a C# application allows you to manage the memory of a computer efficiently as all related data is organized in a structure. It also increases the performance of a C# application.

Common Programming Errors

- Placing the semicolon at the end of structure definition.
- Using the upper case S in the keyword struct.
- Giving initial values to data fields inside struct definition.
- Trying to access a private member outside the struct definition.
- Assigning a structure of one type to a structure different type.
- Attempting to access a struct member by using only the member name.
- Placing the semicolon at the end of enum definition.
- Placing a comma after the last item in the enum list.
- Forgetting to use the dot operator to access enum members.
- In method calls, forgetting to cast **int** type values to **enum** type
- In method calls, passing the values outside the range of valid enum values.

REVIEW QUESTIONS

11.1 State whether the following statements are true or false. If false, correct the statement to make it true

(a) Struct type data are stored on heap.
(b) It is faster to manipulate struct type data compared class type data.
(c) We cannot initialize a data field inside a struct definition.
(d) Structs are implicitly sealed and therefore no class or struct can derive from the struct.
(e) Like classes, structs support inheritance.
(f) By default, the type of an enum is byte.
(g) Integer values can be converted to enum types using casts.
(h) The values of enum type members may be modified during the execution of a program.
(i) Structs can implement multiple interfaces.
(j) C# does not allow structure variables to be members of structures.

11.2 State two significant differences between classes and structs. In what ways are they similar?

11.3 When do we prefer the use of structs over classes?

11.4 Can we initialize variables in a struct?

11.5 Do structs have constructors and destructors? Does a struct support inheritance?

11.6 What is the purpose of a constructor in a structure?

11.7 Why do we use methods as members of a struct?

11.8 What is nesting of structures? Give an example of typical use of nested structures.

11.9 What is enumeration? How is it useful in C# programming?

11.10 Give two examples of typical use of enum type values.

11.11 Find errors, if any, in the following struct definitions and statements:

(a)
```
structure Employee
{
    string name;
    float age;
};
```

(b)
```
struct Point
{
    int x = 10;
    int y = 20;
}
```

(c)
```
struct A
{
    int a;
    A ( int x ) { a = x; }
}
```

(d)
```
struct vector
{
    public int x, y;
    public Vector (int a, int b)
    { x = a, y = b; }
    }
Vector V1 = Vector (10, 20);
Vector V2 = V1;
. . . .
. . . .
```

11.12 Find errors, if any, in the following code segments:

(a)
```
enum XYZ
{
    X,
    Y,
    Z = 100;
}
```

(b)
```
enum Color : byte
{
    Red,
    Blue = 300,
    Green
}
```

(c)
```
enum Color
{
    Red,
    Blue,
    Green = Red
}
```

(d)
```
enum Type
{
    Type 1,
    Type 2
}
Type t1 = 10
    int n = t1
    . . . .
```

11.13 Given these structure definitions and declarations

```
struct Abc
{
    public int a1;
```

```
        public double d1;
    }
    struct Xyz
    {
        public int x1;
        public int x2;
    }
        Abc abc;
        Xyz xyz;
```

find errors, if any, in the following statements:

(a) abc = xyz;
(b) Abc.a1 = 10;
(c) int m = a1 + x1;
(d) int n = xyz.x2+10;

DEBUGGING EXERCISES

11.1 Debug the program given below.

```
using System;
using System.Collections.Generic;
using System.Text;

namespace debugApp1_chap11
{
    enum Students : byte
    {
        passed,
        failed
    }
    struct Stud
    {
        Students stu;
        int id;
    }
    class Program
    {
        static void Main(string[] args)
        {
            Stud st = new Stud();
            st.stu = Students.passed;
            st.id = "S00017_Class5";
            Console.Write("You have ");
            Console.WriteLine(st.stu);
            Console.Write("Your roll number is : ");
            Console.WriteLine(st.id);
        }
    }
}
```

11.2 The program given below demonstrates the use of **public** keyword. Debug the errors so that the program compiles successfully.

```
using System;
using System.Collections.Generic;
using System.Text;
namespace debugApp2_chap11
{
struct StructExample {
    public int m;
    public StructExample(int n) {
    m=n*n;
    }
    void Show() {
            System.Console.WriteLine("This is from Show method in a Struct.");
        }
    }
    class Program
    {
        static void Main(string[] args)
        {
            StructExample st;
            System.Console.WriteLine(st.i);
            st.show();
        }
    }
}
```

11.3 Find errors in the following program for displaying the months using **enum**:

```
using System;
using System.Collections.Generic;
using System.Text;
namespace debugApp3_chap11
{
    enum DateMonth : long
    {
        int oct = 10,
        int nov = 20,
        int dec
    }
    class Program
    {
        static void Main(string[] args)
        {
            long a = DateMonth.oct;
            long b = DateMonth.nov;
            long c = DateMonth.dec;
            Console.WriteLine("October = {0}, November = {1}, December = ", a, b, c);
        }
    }
}
```

PROGRAMMING EXERCISES

11.1 Design a structure data type named Date Of Birth to contain date, month and year of birth. Develop a C# program using this data structure that would assign your date of birth to the individual members and display the date of birth in the following format:

My date of birth is 15/06/75

Do not use any methods in your program.

11.2 Modify the above program using
(a) a constructor to input values to the individual members, and
(b) a method to display the date of birth.

11.3 Design a structure type data using a suitable name for each of the following records:
(a) A student record consisting of name, date of birth, and total marks obtained.
(b) A mailing list consisting of name, door number, street, city and pincode.
(c) An inventory, record consisting of item code, item name, item cost and the total items available.
(d) A book record consisting of the author, title, year of publication and cost.

11.4 Modify the data structure defined in Exercise 11.3(a) as a nested data structure using the date of birth as a structure type data. Develop a suitable C# program to test the nested data structure.

11.5 Develop a program that prompts the user to input the name, door number, street, city and pincode and stores them in the address record designed in Exercise 11.3(b) and display them in an appropriate manner.

11.6 Write a C# program that uses an array of five items of data type designed in Exercise 11.3(c), accept interactively the data of all the five items into the array, and display them in the tabular form as shown below.

Code	Name	Cost	Total items