

# Лекція 3:

## Побудова типів-значень в C#: struct & enum

Клакович Л.М.

# Типи-значення

## ✓ Вбудованя типи:

- Integer types: *long, int, byte*
- floating-point types: *double, float*
- Boolean type: *bool*
- String and characters: *string, char*

## ✓ Типи, визначені користувачем:

- Struct
- Enum

# Перелічуваний тип (Enumeration type)

- **Enum** визначає множину іменованих цілочислових констант, які можна присвоїти змінним.
- За замовчуванням перелічувані константи мають значення 0, 1, 2, ...

```
//Declaration  
enum Days { Sun, Mon, Tue, Wed, Thu, Fri, Sat };
```


```
//Using  
Days today = Days.Mon;  
int dayNumber =(int)today;  
Console.WriteLine("{0} is day number #{1}.", today, dayNumber);
```

**ToString()** відображає ім'я константи



```
string name = Enum.GetName (typeof(Days), Days.Fri);
```

Надає **ім'я** – рядкове представлення перелічуваної константи



```
Days day = (Days) 5;  
Console.WriteLine("{0} is day number #{1}.", day, 5);
```

# Перелічуваний тип (Enumeration type)

- Можна задавати і свої власні значення для перелічуваних констант

```
enum Days {Sat=1, Sun, Mon, Tue, Wed, Thu, Fri};
```

- Кожен тип перелічення має базовий тип, який може бути будь-яким цілим типом, крім [char](#).
- За замовчуванням базовим типом елементів перелічення є [int](#).
- Оголошення іншого цілочислового типу, наприклад [байт](#):

```
enum Days : byte {Sat=1, Sun, Mon, Tue, Wed, Thu, Fri};
```

```
enum Range : long { Max = 2147483648L, Min = 255L };
```

# Перелічуваний тип (Enumeration type)

```
class Program
{
    enum Importance
    {
        None,
        Trivial,
        Regular,
        Important,
        Critical
    };

    static void Main()
    {
        Importance value = Importance.Critical;

        if (value == Importance.Trivial)
        {
            Console.WriteLine("Потім зроблю");
        }
        else if (value == Importance.Critical)
        {
            Console.WriteLine("Негайно до роботи!");
        }
    }
}
```

# Struct – визначає типи-значення

```
struct Point  
{  
    public int X;  
    public int Y;  
};
```

```
static void Main()  
{  
    Point local;  
    local.X = 1;  
    local.Y = 2;  
}
```

# Struct

```
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;

        book1.title = "C# Programming";
        book1.author = "Troelson";
        book1.subject = "C# Programming Tutorial";
        book1.book_id = 649;

        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);
    }
}
```

# Приклад використання структур: DateTime

```
class Program
{
    static void Main()
    {
        DateTime date1 = new DateTime();
        Console.WriteLine(date1);           // 01.01.0001 0:00:00

        DateTime date1 = new DateTime(2018, 2, 20); // рік - місяць - день
        Console.WriteLine(date1); // 20.02.2018 0:00:00

        Console.WriteLine(DateTime.Now);
        Console.WriteLine(date1.DayOfWeek);
        Console.WriteLine(date1.AddHours(3));
    }
}
```



## Типи-значення і типи-посилання

	Типи-значення Value types	Типи-посилання Reference types
Розміщення в пам'яті	в стеку	в керованій купі
Подання змінної	у вигляді локальної копії	у вигляді посилання, що вказує на займане відповідним примірником місце в пам'яті.
Наслідування	від System.ValueType	від будь-яких інших незапечатаних типів (крім ValueType)
Чи може бути базовим для інших типів	ні	так
Передача параметрів методів	за значенням (копія змінної)	За посиланням (адреса змінної)
Чи можна визначати конструктори	Так, але є зарезервований дефолтний конструктор	так
Коли припиняють існування	Коли виходять за межі того контексту, в якому визначалися.	Коли об'єкт піддається збірці сміття.

# Завдання. Struct-Enum

- 1) Визначити перелічення enum `HTTPError`, яке містить назви помилок з присвоєним кодом, наприклад **`BadRequest = 400`**. Ввести ціле число, що представляє код помилки і вивести її назву
- 2) Визначити структуру `Dog` з полями `Name`, `Breed`, `Age`. Визначити змінну `myDog` типу `Dog`, зчитати дані про нього і вивести ім'я на консоль