МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РФ

Брянский государственный технический университет

Утверждаю
Ректор университета
_____О.Н. Федонин
«_____»_____2016 г.

ОБЪЕКТНО-ОРИЕНТИРОВАННОЕ ПРОГРАММИРОВАНИЕ

РАБОТА С ПЕРЕЧИСЛЕНИЯМИ И СТРУКТУРАМИ

Методические указания
к выполнению лабораторной работы
для студентов очной формы обучения
по специальностям 090303 – «Информационная безопасность автоматизированных систем»,
090305 – «Информационно-аналитические системы безопасности»

Брянск 2016

УДК 004.43

Объектно-ориентированное программирование. Работа с перечислениями и структурами: методические указания к выполнению лабораторной работы для студентов очной формы обучения по специальностям 090303 – «Информационная безопасность автоматизированных систем», 090305 – «Информационно-аналитические системы безопасности». – Брянск: БГТУ, 2016. – 12 с.

Разработали:
Ю.А. Леонов, канд. техн. наук, доц.,
Е.А. Леонов, канд. техн. наук, доц.

Рекомендовано кафедрой «Компьютерные технологии и системы» БГТУ (протокол № 2 от 20.01.16)

Методические указания публикуются в авторской редакции

Научный редактор Е.А. Леонов
Компьютерный набор Ю.А. Леонов

Темплан 2016 г., п.

Подписано в печать Формат 60х84 1/16. Бумага офсетная. Офсетная печать.Усл. печ. л. 0,7 Уч. – изд. л. 0,7 Тираж 1 экз. Заказ Бесплатно

Издательство Брянского государственного технического университета
241035, Брянск, бульвар 50 лет Октября, 7, БГТУ, тел. 58-82-49
Лаборатория оперативной полиграфии БГТУ, ул. Институтская, 16

1. ЦЕЛЬ РАБОТЫ

Целью лабораторной работы является приобретение практических навыков при работе с перечислениями и структурами данных.

Продолжительность лабораторной работы – 4 ч.

2. КРАТКИЕ ТЕОРЕТИЧЕСКИЕ СВЕДЕНИЯ

Данные, с которыми работает программа, хранятся в оперативной памяти. Компилятору необходимо точно знать, сколько места они занимают, как именно закодированы и какие действия с ними можно выполнять. Все это задается при описании данных с помощью типа.

Все типы данных можно условно разделить на встроенные и пользовательские. *Встроенные типы* не требуют предварительного определения. Для каждого типа существует ключевое слово, которое используется при описании переменных, констант и т.д. Характеристики встроенных типов менять нельзя.

Пользовательские типы данных позволяют менять структуру и характеристики базовых классов, на которых они основаны.

Все пользовательские типы в платформе .NET Framework регламентируются системой общих типов (CTS – common type system) и делятся на *типы значений* и *ссылочные типы* (рис. 1).

Типы значений – это типы данных, объекты которых содержат фактические значения. Если экземпляр типа значения присваивается переменной, то эта переменная получает новую копию значения.

Ссылочные типы – это типы данных, объекты которых представлены ссылкой (аналогичной указателю) на фактическое значение объекта. Если экземпляр ссылочного типа присваивается переменной, то эта переменная будет ссылаться (указывать) на исходное значение. Копирования при этом не происходит.

Система общих типов CTS определяет способ объявления, использования и управления типами в среде CLR, а также является важной составной частью поддержки межъязыковой интеграции в среде выполнения. Система общих типов выполняет следующие функции:

- формирует инфраструктуру, которая позволяет обеспечивать межъязыковую интеграцию, безопасность типов и высокопроизводительное выполнение кода;
- предоставляет объектно-ориентированную модель, поддерживающую полную реализацию многих языков программирования;

- определяет правила, которых необходимо придерживаться в языке. Эти правила помогают обеспечить взаимодействие объектов, написанных на разных языках;
- предоставляет библиотеку, которая содержит типы-примитивы (например, Boolean, Byte, Char, Int32 и UInt64), используемые в разработке приложений.

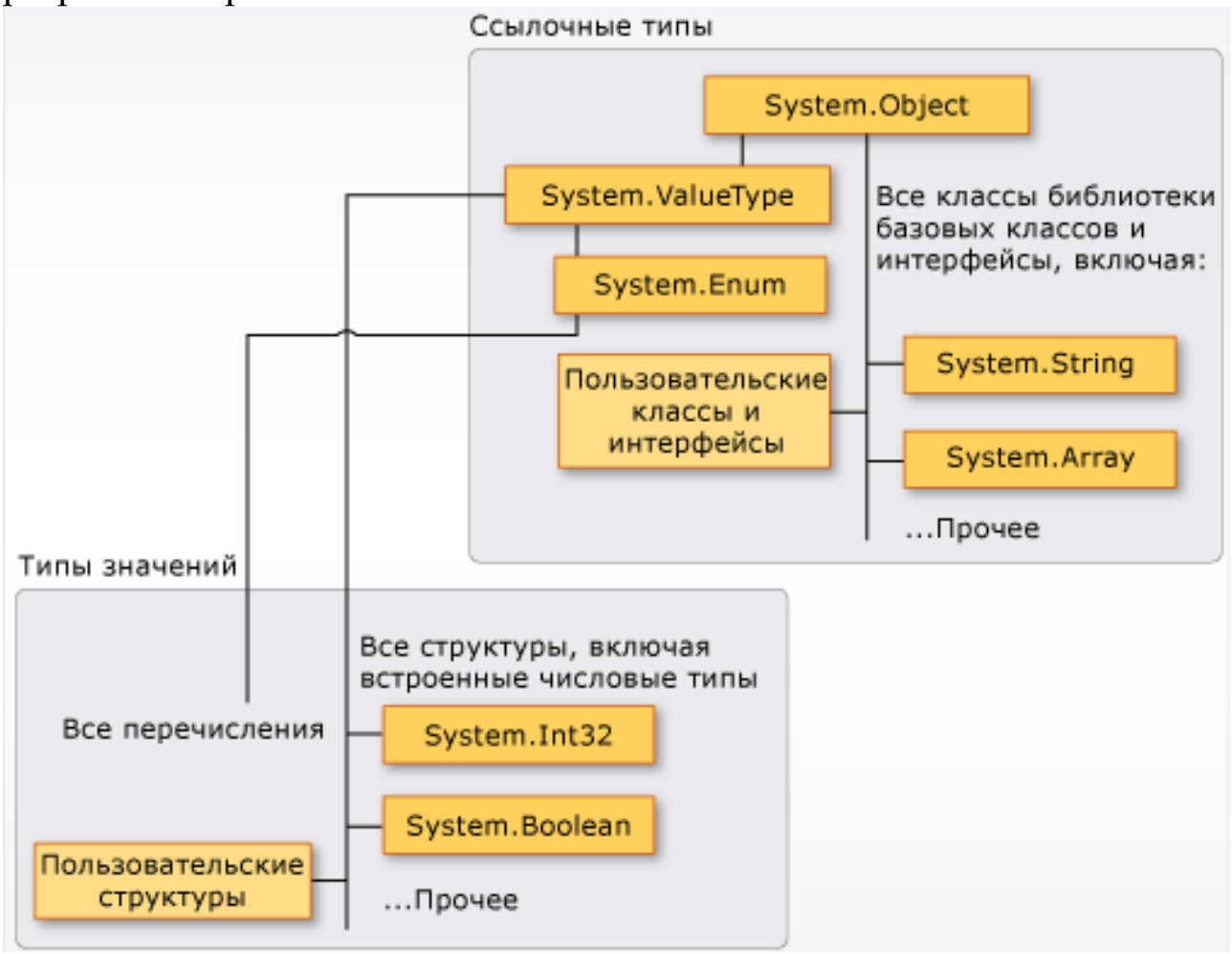

Рис. 1. Общая классификация типов данных

Система общих типов в платформе .NET Framework поддерживает следующие пять категорий типов: перечисления, классы, структуры, интерфейсы, делегаты. Рассмотрим подробно перечисления и структуры.

2.1. Работа с перечислениями

C# позволяет создать собственный набор именованных констант с помощью ключевого слова enum. Такие типы данных позволяют объявить набор имен или других значений литералов, определяющих все возможные значения, которые могут быть назначены переменной.

Например, если в программе ведется работа с днями недели, может потребоваться создать новый тип с именем DayOfWeek. Затем можно объявить новую переменную типа DayOfWeek, а затем присвоить ей значение. Использование этого типа данных способствует повышению удобочитаемости кода, кроме того, снижается вероятность назначения переменной недопустимого или неожиданного значения.

Пример.

```
public enum DayOfWeek
 {
     Sunday = 0,
     Monday = 1,
     Tuesday = 2,
     Wednesday = 3,
     Thursday = 4,
     Friday = 5,
     Saturday = 6
 }

 class Program
 {
     static void Main()
     {
         DayOfWeek day = DayOfWeek.Monday;
         int i = (int)DayOfWeek.Monday;

         Console.WriteLine(day);
         Console.WriteLine(i);
     }
 }
```

По умолчанию базовым типом каждого элемента перечисления является тип int. Можно задать другой целочисленный тип, используя двоеточие, как показано в следующем примере.

```
enum Months : byte { Jan, Feb, Mar, Apr, May, Jun, Jul, Aug, Sep,
Oct, Nov, Dec };
```

При объявлении перечисления в качестве базового типа могут использоваться следующие типы: byte, sbyte, short, ushort, int, uint, long или ulong.

Элементам списка перечислителя типа перечисления можно присвоить любые значения, и можно также использовать вычисленные значения:

```
enum MachineState
{
    PowerOff = 0,
    Running = 5,
    Sleeping = 10,
```

```
    Hibernating = Sleeping + 5
}
```

Тип перечисления можно использовать для определения битовых флагов, благодаря чему экземпляр типа перечисления может хранить любую комбинацию значений, определенных в списке перечислителя. (некоторые комбинации могут не иметь смысла или могут быть недопустимы в коде программы).

Перечисление битовых флагов создается применением атрибута System.FlagsAttribute и определением соответственным образом значений, так чтобы на них могли выполняться битовые операции AND, OR, NOT и XOR. В перечисление битовых флагов включите именованную константу с нулевым значением, что означает «флаги не установлены». Не придавайте флагу нулевое значение, если оно не означает «флаги не установлены».

В следующем примере определена другая версия перечисления Days, имеющая имя Days2. У Days2 имеется атрибут Flags и каждому значению присваивается следующая степень числа 2. Это позволяет создать переменную Days2 со значением Days2.Tuesday и Days2.Thursday.

```
[Flags]
enum Days2
{
    None = 0x0,
    Sunday = 0x1,
    Monday = 0x2,
    Tuesday = 0x4,
    Wednesday = 0x8,
    Thursday = 0x10,
    Friday = 0x20,
    Saturday = 0x40
}
class MyClass
{
    Days2 meetingDays = Days2.Tuesday | Days2.Thursday;
}
```

Чтобы установить флаг на перечислении, используйте побитовый оператор OR, как показано в следующем примере:

```
// Initialize with two flags using bitwise OR.
meetingDays = Days2.Tuesday | Days2.Thursday;

// Set an additional flag using bitwise OR.
meetingDays = meetingDays | Days2.Friday;

Console.WriteLine("Meeting days are {0}", meetingDays);
```

```
// Output: Meeting days are Tuesday, Thursday, Friday

// Remove a flag using bitwise XOR.
meetingDays = meetingDays ^ Days2.Tuesday;
// Output: Meeting days are Thursday, Friday
 Console.WriteLine("Meeting days are {0}", meetingDays);
```

Чтобы определить, установлен ли конкретный флаг, используйте побитовый оператор AND, как показано в следующем примере.

```
// Test value of flags using bitwise AND.
bool test = (meetingDays & Days2.Thursday) == Days2.Thursday;
Console.WriteLine("Thursday {0} a meeting day.", test == true ? "is"
: "is not");
// Output: Thursday is a meeting day.
```

Все перечисления являются экземплярами типа System.Enum. Нельзя произвести новые классы от класса System.Enum, но можно использовать его методы для получения данных экземпляра перечисления и для обработки этих данных.

```
string s = Enum.GetName(typeof(Days), 4);
Console.WriteLine(s);

Console.WriteLine("The values of the Days Enum are:");
foreach (int i in Enum.GetValues(typeof(Days)))
    Console.WriteLine(i);

Console.WriteLine("The names of the Days Enum are:");
foreach (string str in Enum.GetNames(typeof(Days)))
  Console.WriteLine(str);
```

2.2. Работа со структурами

Тип struct – это тип значения, который обычно используется для инкапсуляции небольших групп связанных переменных, например, координат прямоугольника или характеристик предмета в инвентаре. В следующем примере показано простое объявление структуры:

```
public struct Book
{
    public decimal price;
    public string title;
    public string author;
}
```

Структуры также могут содержать конструкторы, константы, поля, методы, свойства, индексаторы, операторы, события и вложенные типы, хотя, если требуется несколько таких членов, тип необходимо заменить классом.

Структуры могут реализовать интерфейс, но не могут наследоваться из другой структуры. По этой причине члены структуры не могут объявляться как protected.

Тип struct подходит для создания несложных объектов, таких как Point, Rectangle и Color. Хотя point удобно представить в виде класса с автоматически реализуемыми свойствами, в некоторых сценариях структура может оказаться более эффективной.Например, при объявлении массива из 1000 объектов Point потребуется выделить дополнительную память для хранения ссылок на все эти объекты, и структура в таком случае будет более экономичным решением.

```
public struct Point
{
    public int x, y;

    public Point(int x, int y)
    {
        this.x = x;
        this.y = y;
    }
}
```

Определение конструктора по умолчанию (без параметров) для структуры является ошибкой. Ошибкой также является инициализация поля экземпляра в основной части структуры. Можно инициализировать члены структуры с помощью параметризованного конструктора или путем доступа к членам по отдельности после объявления структуры. Любые закрытые или в ином случае недоступные члены можно инициализировать только в конструкторе.

При создании объекта структуры с помощью оператора new объект создается и вызывается соответствующий конструктор. В отличие от классов структуры можно создавать без использования оператора new. В таком случае вызов конструктора отсутствует, что делает выделение более эффективным. Однако поля остаются без значений и объект нельзя использовать до инициализации всех полей.

Когда структура содержит ссылочный тип в качестве члена, конструктор по умолчанию члена должен вызываться явно, в противном случае член останется без значений и структура не сможет использоваться.

В отличие от классов структуры не поддерживают наследование. Структура не может быть унаследованной от другой структуры

или класса и не может быть основой для других классов. Однако структуры наследуют от базового класса Object. Структуры могут реализовывать интерфейсы; этот механизм полностью аналогичен реализации интерфейсов для классов.

Класс нельзя объявить с помощью ключевого слова struct. В C# классы и структуры семантически различаются. Структура является типом значения, тогда как класс – это ссылочный тип.

Если не требуется использовать семантику ссылочных типов, небольшие классы могут более эффективно обрабатываться системой, если их объявить, как структуры.

Структуры используют большую часть того же синтаксиса, что и классы, однако они более ограничены по сравнению с ними. Приведем основные особенности структур:

- в объявлении поля структуры не могут быть инициализированы за исключением случаев, когда они объявлены как постоянные или статические;
- структура не может объявлять используемый по умолчанию конструктор (конструктор без параметров) или деструктор;
- структуры копируются при присваивании; при присваивании структуры к новой переменной выполняется копирование всех данных, а любое изменение новой копии не влияет на данные в исходной копии;
- структуры являются типами значений;
- в отличие от классов, структуры можно создавать без использования оператора new;
- структуры могут объявлять конструкторы, имеющие параметры;
- структура не может быть унаследованной от другой структуры или класса и не может быть основой для других классов; все структуры наследуют непосредственно от System.ValueType, который наследует от System.Object;
- структуры могут реализовывать интерфейсы;
- структура может использоваться как тип, допускающий значение null, и ей можно назначить значение null.

3. ПОРЯДОК ВЫПОЛНЕНИЯ РАБОТЫ

Для выполнения лабораторной работы необходимо предварительно ознакомиться с краткими теоретическими сведениями, приводимыми в методических указаниях, а при необходимости с литературными источниками, приводимыми в списке рекомендуемой литературы.

В соответствии с заданием написать две программы на языке C#, которые демонстрируют работу:

1) с перечислениями;
2) с структурами.

Общая формулировка задания для работы с перечислениями

Необходимо выполнить следующие операции с перечислениями:

1) описать перечисление согласно варианту (табл. 1);
2) объявить переменную перечисляемого типа данных;
3) инициализировать переменную значением с клавиатуры;
4) вывести все значения перечисляемого типа данных на экран при этом введенное с клавиатуры значение подсветить другим цветом.

Общая формулировка задания для работы со структурами

Необходимо выполнить следующие операции со структурами:

1) описать структуру согласно варианту (табл. 2);
2) объявить переменную структурного типа;
3) описать конструктор, инициализирующий поля структуры;
4) описать метод PrintStruct, который выводит на экран значения полей структуры в формате: <название поля> - <значение>;
5) инициализировать поля структуры значениями, введенными с клавиатуры;
6) вывести поля структуры на экран с помощью PrintStruct.

4. ЗАДАНИЯ К РАБОТЕ

4.1. Задания на работу с перечислениями

Таблица 1

№ варианта	Состав перечисления
1	Кнопки компьютерной мышки (левая, центральная, правая и т.д.)
2	Вид телефонного звонка (входящий, исходящий и т.д.)

№ варианта	Состав перечисления
3	Вид мебели (стул, стол и т.д.)
4	Цвета светофора (красный, желтый, зеленый)
5	Весовые категории в спорте (легкий, тяжелый и т.д.)
6	Агрегатные состояния вещества (газ, жидкость и т.д.)
7	Виды родственников (муж, жена, брат, дед и т.д.)
8	Виды сыров (плавленый, твердый, с плесенью т.д.)
9	Воинские звания сухопутных частей вооруженных сил РФ (рядовой, ефрейтор, мл. сержант и т.д.)
10	Воинские звания морских частей вооруженных сил РФ (матрос, ст. матрос и т.д.)
11	Виды стрелкового оружия (пистолет, автомат и т.д.)
12	Виды белокурых волос (белобрысые, льняные и т.д.)
13	Виды цветов волос (брюнет, блондин, рыжий и т.д.)
14	Виды жанров кино (комедия, мелодрама, боевик и т.д.)
15	Виды архитектурных стилей (барокко, модерн, конструктивизм и т.д.)
16	Виды изобразительного искусства (портрет, иконография, живопись и т.д.)
17	Виды музыкального направления «блюз» (классический, ритм-н-блюз, соул, фанк и т.д.)
18	Виды музыкального направления «джаз» (бибоп, биг-бэнд, кул-джаз и т.д.)
19	Виды альтернативной музыки (брит-поп, гранж, дрим-поп, инди и т.д.)
20	Виды спортивных игр с мячом (волейбол, баскетбол, гандбол и т.д.)

4.2. Задания на работу со структурами

Таблица 2

№ варианта	Вид структуры
1	Книга
2	Принтер
3	Телевизор
4	Автомобиль
5	Пылесос
6	Электрический чайник
7	Квартира
8	Струнный музыкальный инструмент
9	Клавишный музыкальный инструмент
10	Духовой музыкальный инструмент

№ варианта	Вид структуры
11	Ткань
12	Дерево
13	Мост
14	Подъемный кран
15	Бумага
16	Напольное покрытие
17	Человек
18	Лекарство
19	Самолет
20	Холодильник

5. КОНТРОЛЬНЫЕ ВОПРОСЫ

1. Какие типы данных называют пользовательскими?
2. Чем отличаются типы значений и ссылочные типы?
3. Какой тип данных называют перечислением и для чего он используется?
4. Какой тип данных называют структурой и для чего он используется?
5. Опишите синтаксис объявления конструктора структуры.
6. Каким образом используются перечисления для работы с битовыми флагами?
7. Перечислите основные особенности структур.
8. Опишите синтаксис объявления перечисления.
9. Опишите синтаксис объявления структуры.

6. СПИСОК РЕКОМЕНДУЕМОЙ ЛИТЕРАТУРЫ

1. Павловская, Т.А. C#. Программирование на языке высокого уровня. – Изд.: Питер, 2009. – 432с.
2. Троелсен, Э. Язык программирования C# 2010 и платформа .NET 4. – Изд.: Вильямс, 2011. – 1392с.
3. Нейгел, К., C# 4.0 и платформа .NET 4 для профессионалов / Билл Ивьен, Джей Глинн, Карли Уотсон, Морган Скиннер. – Изд.: Питер, 2011. – 1440с.
4. Джесс Либерти. Программирование на C#. – Изд.: КноРус, 2003. – 688с.
5. Харви Дейтел. C# в подлиннике. Наиболее полное руководство. – Изд.: БХВ-Петербург, 2006. – 1056с.