Практическаяработа № 1.7

Тема: Перечисления. Структуры. Интерфейсы

Цель: Научиться разрабатывать интерфейсыв C# и раскрывать их в классах.Научиться использовать структуры и перечисления в C#.

Время выполнения:6 часов.

Обеспечение занятия:персональный компьютер, среда разработки VisualStudio2010.

Методические рекомендации:

3) Троелсен Э. Язык программирования C# 2010 и платформа .NET 4.0, 5-е изд. : Пер. с англ. — М.: ООО "И.Д. Вильямс", 2011.

6) Андрианова А.А., Исмагилов Л.Н., Мухтарова Т.М. Практикум по курсу «Объектно-ориентированное программирование» на языке C#: Учебное пособие / А.А. Андрианова, Л.Н. Исмагилов, Т.М. Мухтарова. – Казань: Казанский (Приволжский) федеральный университет, 2012 (стр. 46)

Содержание и последовательность выполнения работы

I. Входной контроль.

1) Наследование, полиморфизм в C#. [1, 2].
2) Интерфейсы в C#[1, 2].

II. Задания для выполнения работы.

Задание №1. *Работа с перечислениями и структурами в C#*

Разработать программу, в которой выполняется ввод списка записей определенного типа (описание типа приведено после таблицы 1.7.1), а затем – обработка списка (таблица 1.7.1). Сначала в программе должен вводиться размер списка (целое число), а сам список создается в виде массива структур в динамической памяти. Продемонстрировать работу с перечислениями. Для этого дополнить исходную структуру полем типа перечисление.

Задание №2. *Разработка интерфейса для работы с математическими объектами.*

Разработать интерфейс работы с математическими объектами IMathObject и определить в нем общие операции: сложение, вычитание, умножение на объект и умножение на число. Также включить в интерфейс метод получения строкового представления объекта, чтобы использовать полученную строку для вывода объекта. [6, стр. 46]

Задание №3 *Раскрытие интерфейса и реализация классов*

Раскрыть интерфейс IMathObject на примере классов согласно вариантам задания (таблица 1.7.2). Для массива объектов, которые раскрывают интерфейс IMathObject, создать метод-обобщение, который выполняет операцию согласно индивидуальному варианту задания.

III. Краткие теоретические сведения и примеры программного кода

По заданию №3 примеры программного кода см. в [6, стр.46].

Структуры - это составной объект, в который входят элементы любых типов, в том числе и функций. В отличие от массива, который является однородным объектом, структура может быть неоднородной, т.е. структура объединяет несколько переменных разного типа данных. Переменные, которые объединены структурой, называются членами, элементами или полями структуры.

Синтаксис структуры:

[атрибуты] [спецификаторы] struct имя_структуры [: интерфейсы] тело_структуры [;]

Из *спецификаторов доступа* допускаются только public и private (последний – только для вложенных структур).

Тело структуры может состоять из констант, полей, методов, свойств, событий, индексаторов, операций, конструкторов и вложенных типов.

Объявление структуры является оператором, и поэтому в конце должна стоять точка с запятой. *При этом никакая переменная не объявлена*. Выделения памяти под переменную не произошло.

При объявлении структуры задан так называемый *шаблон структуры* и **определен новый пользовательский тип данных.**

Пример 1

```
/*Определения структуры студент.
Обратите внимание на перегруженный метод ToString: он позволяет выводить экземпляры структуры (переменные типа структуры) на экран, поскольку неявно вызывается в методе Console.WriteLine. */
namespace ConsoleApplication
{
//объявление структуры, обратите внимание на место в консольном приложении
public struct student
{
//поля структуры
public string name;
public int kurs;
public string gruppa;
public int stipendia;
// метод структуры (перегруженный)
public override string ToString()
{
return (string.Format( "Имя студента {0}; Курс{1}; Группа № {2};Размер стипендии: {3}" , name, kurs,gruppa,stipendia ) );
}//конец метода
}; //конец описания структуры student
class Program
{
static void Main(string[] args)
{
student s; //объявление экземпляра (переменной) структуры
Console.WriteLine("Введите данныео студенте:");
Console.WriteLine("Имя:");
//ДОСТУП к элементам структуры – через операцию "точка"
s.name = Console.ReadLine();
Console.WriteLine("Курс:");
s.kurs = Convert.ToInt32(Console.ReadLine());
Console.WriteLine("Группа");
s.gruppa = Console.ReadLine();
Console.WriteLine("Размер стипендии:");

s.stipendia = Convert.ToInt32(Console.ReadLine());
// вызов перегруженного метода ToString()
 Console.WriteLine( "structura s: " +s); //вывод всех полей структуры на экран
Console.ReadKey();
```

```
} }}
```

При выводе **экземпляра структуры на экран выполняется *упаковка,*** то есть неявное преобразование в ссылочный тип. Упаковка применяется в других случаях, когда структурный тип используется там, где ожидается ссылочный, например, при преобразовании экземпляра структуры к типу реализуемого ею интерфейса. **При обратном преобразовании – из ссылочного типа в структурный – выполняется *распаковка*.**

При присваивании структур создается копия значений полей. То же самое происходит и при передаче структур в качестве параметров по значению. Для экономии ресурсов ничто не мешает передавать структуры в методы по ссылке с помощью ключевых слов ref или out.

В следующем примере приведен пример описания структуры, представляющей комплексное число.

Пример 2

```
namespace ConsoleApplication1
{
struct Complex
{
public double re, im; //поля структуры – действительная и мнимая часть

// особый метод структуры - конструктор
public Complex( double re_, double im_)
{
re = re_; im = im_; //присваивание полям структуры аргументов метода
}
// перегруженный метод структуры - операция сложения двух комплексных
чисел
public static Complex operator + ( Complex a, Complex b)
{
return new Complex( a.re + b.re, a.im + b.im );
}
//перегруженный метод преобразования данных в строку
public override string ToString()
{
return ( string.Format( "({0,2:0.##};{1,2:0.##})" , re, im ) );
}
}
class Class1
{ static void Main()
{
// объявление переменной структуры и вызов конструктора(метода)
Complex a = new Complex( 1.2345, 5.6 ) ;
Console.WriteLine( "a = " + a);
Complex b;
b.re = 10; b.im = 1;
Console.WriteLine( "b = " + b);
Complex c = new Complex();
Console.WriteLine( "c = " + c);
c = a + b;
Console.WriteLine( "c = " +c);
} }}
```

```
Результат работы программы:
a = (1,23; 5,6)
```

```
b = (10; 1)
c = ( 0; 0)
c = (11,23; 6,6)
```

Особенно значительный выигрыш в эффективности можно получить, используя массивы структур.

Пример описания массива структур:

```
//описание структуры см. в предыдущем примере
Complex [] mas = new Complex[4];
for ( int i = 0; i < 4; ++i )
{
mas[i].re = i;
mas[i].im = 2 * i;
}
// вывод на экран всех элементов массива структур
foreach ( Complex elem in mas ) Console.WriteLine( elem ); …
```

Перечисления

При написании программ часто возникает потребность определить несколько связанных между собой именованных констант, при этом их конкретные значения могут быть не важны. Для этого удобно воспользоваться **перечисляемым типом данных**, *все возможные значения которого задаются списком целочисленных констант,* например:

enum Menu { Read, Write, Append, Exit }

enum Радуга { Красный, Оранжевый, Желтый, Зеленый, Синий, Фиолетовый }

Для каждой константы задается ее символическое имя. По умолчанию константам присваиваются последовательные значения типа int, начиная с 0, но можно задать и собственные значения, например:

enum Nums { two = 2, three, four, ten = 10, eleven, fifty = ten + 40};

Константам three и four присваиваются значения 3 и 4, константе eleven – 11. Имена перечисляемых констант внутри каждого перечисления должны быть уникальными, а значения могут совпадать.

Преимущество перечисления перед описанием именованных констант состоит в том, что связанные константы нагляднее; кроме того, компилятор выполняет проверку типов, а интегрированная среда разработки подсказывает возможные значения констант, выводя их список.

Операции с перечислениями

С переменными перечисляемого типа можно выполнять:

- арифметические операции (+, –, ++, – –)
- логические поразрядные операции (^, &, |, ~)
- сравнить их с помощью операции отношения (<, <=, >, >=, = =, !=)
- получать размер в байтах (sizeof).

При использовании переменных перечисляемого типа в целочисленных выражениях и операциях присваивания требуется явное *преобразование типа*. Переменной перечисляемого типа можно присвоить *любое значение*, представимое с помощью базового типа, то есть не только одно из значений, входящих в тело перечисления. Присваиваемое значение становиться новым элементом перечисления.

Пример 3

```
namespace ConsoleApplication10
{
struct Боец
{
public enum Воинское_Звание
{
Рядовой, Сержант, Лейтенант, Майор, Полковник, Генерал
}
public string Фамилия;
public Воинское_Звание Звание;
}
class Program
{
static void Main(string[] args)
{
Боец x;
x.Фамилия = "Иванов";
x.Звание = Боец.Воинское_Звание.Сержант;
for (int i = 1976; i < 2006; i += 5)
{
if (x.Звание < Боец.Воинское_Звание.Генерал) ++x.Звание;
Console.WriteLine("Год: {0} {1} {2}", i, x.Звание, x.Фамилия);
}
Console.ReadKey(); } }}
```

Результат работы программы:

Год: 1976 Лейтенант Иванов

Год: 1981 Майор Иванов

Год: 1986 Полковник Иванов

Год: 1991 Генерал Иванов

Год: 1996 Генерал Иванов

Год: 2001 Генерал Иванов

Базовый класс System.Enum

Все перечисления в C# являются потомками базового класса System.Enum, который снабжает их некоторыми полезными методами. Статический метод **GetName** позволяет получить символическое имя константы по ее номеру, например:

Console.WriteLine (Enum.GetName(typeof(Боец.Воинское_Звание) , 1)); // сержант

Операция typeof возвращает тип своего аргумента

Статические методы **GetNames и GetValues** формируют, соответственно, массивы имен и значений констант, составляющих перечисление.

Пример 4

```
namespace ConsoleApplication10
{
```

```
struct Боец
{
public enum Воинское_Звание
{
Рядовой, Сержант, Лейтенант, Майор, Полковник, Генерал
}
public string Фамилия;
public Воинское_Звание Звание;
}
class Program
{
static void Main(string[] args)
{
Боец x;

Array names = Enum.GetNames( typeof(Боец.Воинское_Звание) );
Console.WriteLine("Количество элементов в перечислении:" + names.Length
);
foreach ( string elem in names ) Console.Write( " " + elem );
Array values = Enum.GetValues( typeof(Боец.Воинское_Звание) );
foreach ( Боец.Воинское_Звание elem in values )
Console.Write( " " + (byte) elem );
Console.ReadKey(); } }}
```

Статический метод **IsDefined** возвращает значение true, если константа с заданным символическим именем описана в указанном перечислении, и false в противном случае, например:

if (Enum.IsDefined(typeof(Боец.Воинское_Звание), "Сержант"))

Console.WriteLine("Константа с таким именем существует");

else Console.WriteLine("Константа с таким именем не существует");

Статический метод **GetUnderlyingType** возвращает имя базового типа, на котором построено перечисление. Например, для перечисления Боец.Воинское_Звание будет получено System.Int32:

Console.WriteLine(Enum.GetUnderlyingType(typeof(Боец.Воинское_Звание)));

Рекомендации по программированию

Область применения *структур* – типы данных, имеющие небольшое количество полей, с которыми удобнее работать как со значениями, а не как со ссылками. Накладные расходы на динамическое выделение памяти для экземпляров небольших классов могут весьма значительно снизить быстродействие программы, поэтому их эффективнее описывать как структуры.

Преимущество использования *перечислений* для описания связанных между собой значений состоит в том, что это более наглядно и инкапсулировано, чем россыпь именованных констант. Кроме того, компилятор выполняет проверку типов, а интегрированная среда разработки подсказывает возможные значения констант, выводя их список.

Пример 5 программы с использованием структур с методами и перечислений

```
using System;
using System.Collections.Generic;
using System.Text;
//Багги - если неверно вводится знак зодиака, то по умолчанию буде Овен
```

```
// если вводятся одинаковые фамилии
namespace ConsoleApplication10
{
struct Человек
{
public enum Знак_Зодиака
{
Овен,Скорпион,Рак,Стрелец,Рыбы,Весы,Близнецы,Дева,Козерог,Водолей,Лев
}
public string Фамилия;
public Знак_Зодиака Знак;
public override string ToString()
{
return (string.Format("Фамилия {0}. Знак зодиака {1}", Фамилия,Знак));
}//конец метода

public void vvod()
{
Console.WriteLine("Введите фамилию человека:");
Фамилия=Console.ReadLine();
Console.WriteLine("Введите знак зодиака:");
string buf =Console.ReadLine();
switch (buf)
{
case "Овен" : Знак = Знак_Зодиака.Овен; break;
case "Скорпион": Знак = Знак_Зодиака.Скорпион; break;
case "Рак": Знак = Знак_Зодиака.Рак; break;
case "Стрелец": Знак = Знак_Зодиака.Стрелец; break;
case "Рыбы": Знак = Знак_Зодиака.Рыбы; break;
case "Весы": Знак = Знак_Зодиака.Весы; break;
case "Близнецы": Знак = Знак_Зодиака.Близнецы; break;
case "Дева": Знак = Знак_Зодиака.Дева; break;
case "Козерог": Знак = Знак_Зодиака.Козерог; break;
case "Водолей": Знак = Знак_Зодиака.Водолей; break;
case "Лев": Знак = Знак_Зодиака.Лев; break;
// default: Console.WriteLine("Значение не определено"); break;
}
}

}
class Program
{
static void Main(string[] args)
{
Console.WriteLine("Введите количество человек:");
int n = Convert.ToInt32(Console.ReadLine());
Человек[] x=new Человек[n];
for (int i = 0; i < n; i++)
x[i].vvod();
Console.WriteLine("Информация в базе:");
for (int i = 0; i < n; i++)
Console.WriteLine(x[i]);
Console.WriteLine("Введите фамилию для поиска знака зодиака:");
string fam=Console.ReadLine();
int k=0;
for (int i = 0; i < n; i++)
```

```
{
if (x[i].Фамилия == fam) { Console.WriteLine(x[i]); k++; break; }
}
if (k == 0) Console.WriteLine("Такой фамилии нет");

Console.ReadKey();
}
}
}
```

IV. Выходнойконтроль.

Разработанные вами приложения согласно индивидуальным вариантам задания продемонстрировать преподавателю. Оформить отчет, в котором указатьвариантиндивидуального задания, формулировка задания, листинги программ с подробными комментариями, оформленные согласно соглашений по оформлению исходного кода, результаты работы программы в виде скриншотов, общие выводы по работе. Пример титульного листа и листа отчета на каждый день – смотри в moolde.

Преподаватель: _____ Бровко Н.В.

ВАРИАНТЫ ИНДИВИДУАЛЬНЫХ ЗАДАНИЙ

Таблица 1.7.1 – Индивидуальные варианты к заданиям №1

№	Структура	Сортировка	Обработка
1	СТУДЕНТ	Сортировать по курсу, а внутри курса по ФИО	Вывести все записи о студентах определенной специальности
2	КНИГА	Упорядочить по автору, а внутри автора по количеству страниц	Вывести на экран записи о книгах, изданных за определенный период
3	МАШИНА	Упорядочить список по марке, а внутри марки по году выпуска	Вывести на экран записи о машинах определенного цвета
4	ФАЙЛ	Упорядочить файлы по дате создания, а внутри даты по имени	Вывести на экран все файлы, размер которых лежит в заданном интервале
5	ПРОЦЕССОР	Упорядочить записи по производителю, а внутри производителя по частоте	Вывести на экран все записи определенной модели
6	САМОЛЕТ	Упорядочить по марке модели, а внутри по количеству часов налета	Вывести все записи о самолетах определенного типа
7	БИЛЕТ	Упорядочить билеты по дате, а внутри даты – по времени сеанса	Вывести на экран все записи о билетах, номер ряда которых принадлежит заданному набору
8	СООБЩЕНИЕ	Упорядочить по номеру устройства, а внутри – по тексту сообщения	Вывести на экран сообщения, полученные за определенный временной период
9	НАКЛАДНАЯ	Упорядочить по дате, а внутри даты по сумме накладной	Вывести на экран все накладные с определенным ответственным лицом
10	ФИРМА	Упорядочить по ФИО владельца, а внутри по дате основания	Вывести все записи, в названии которых встречается определенное слово
11	МАРШРУТ	Упорядочить по станции отправления, а внутри по станции прибытия.	Вывести все маршруты с временем следования больше определенного количества минут
12	ТЕЛЕФОН	Упорядочить по тарифу, а внутри тарифа по фамилии владельца	Вывести на экран все записи с определенной комбинацией из двух цифр
13	ВИДЕОПОИСК	Упорядочить по кинокомпании, а внутри – по продолжительности	Вывести на экран все записи о фильмах определенного жанра
14	ПОЕЗД	Упорядочить по станции отправления, а внутри по номеру поезда	Вывести на экран все записи, количество вагонов в которых лежит в заданном диапазоне
15	УСТРОЙСТВО	Упорядочить по типу устройства, а внутри по фирме производителю	Вывести на экран все записи об устройствах, выпущенных в определенный период.

СТУДЕНТ:
фамилия, имя, отчество (строки по 15 символов);
специальность (строка из двух символов);
курс, на котором учится студент (целое число от 1 до 5);
дата рождения (в формате дд.мм.гггг).
Формат ввода: Фамилия Имя Отчество [Специальность, Курс], Дата рождения

КНИГА:
код ISBN (строка 15 символов)
фамилия и инициалы автора (строка 15 символов);
название книги (строка 20 символов);

год издания (целое четырехзначное число);
количество страниц (целое четырехзначное число).
Формат ввода: код ISBN: Фамилия И. О. «Название», год - количество страниц

МАШИНА:
марка (строка 15 символов);
модель (строка 5 символов);
серийный номер (целое семизначное число);
год выпуска (целое четырехзначное число);
цвет (трехзначный целочисленный код)
Формат ввода: Марка Модель [Серийный номер], Год выпуска, Цвет

ФАЙЛ:
имя файла (строка 30 символов, уникальное поле);
размер файла (целое число);
дата создания (в формате дд.мм.гггг);
время создания (в формате чч:мм).
Формат ввода: Имя файла, размер файла, дата и время создания

ПРОЦЕССОР:
производитель (строка 10 символов);
модель (строка 15 символов);
тактовая частота в MHz (целое число меньшее 10000);
размеры КЭШ памяти для данных и команд в KB (целые числа меньше 32000).
Формат ввода: Производитель Модель (Тактовая частотаMHz, КЭШДанныхKB, КЭШКомандKB)

САМОЛЕТ:
серийный номер (целое восьмизначное число)
марка – модель (строка 20 символов);
год выпуска (целое четырехзначное число);
признак (P – пассажирский, C – грузовой);
время налета (целое число, в часах, меньшее 15000).
Формат ввода: Серийный номер Марка – модель (Признак), Год выпуска, Время налета

БИЛЕТ:
название сеанса (строка 30 символов);
дата и время сеанса (в формате дд.мм.ггггчч:мм);
номер ряда (целое число меньше 50);
номер места (целое число меньше 50).
Формат ввода: Номер «Название сеанса» Дата и время, Номер ряда, Номер места

СООБЩЕНИЕ:
номер устройства (целое трехзначное число)
ID сообщения (целое восьмизначное число в 16-ричной системе);
текст (строка 30 символов);
дата и время отправления (в формате дд.мм.ггггчч:мм).
Формат ввода: Номер устройства: ID сообщения «Текст» Дата и время отправления

НАКЛАДНАЯ:
номер накладной (целое число, уникальное поле);
дата накладной (в формате дд.мм.гггг);
общая сумма по накладной (вещественное число);
ФИО ответственного лица (строка 20 символов).
Формат ввода: Номер: «Дата», Сумма, ФИО ответственного лица

ФИРМА:
название (строка 20 символов);
УНН (целое десятизначное число, уникальное поле);
ФИО владельца (строка 20 символов);

дата основания (в формате дд.мм.гггг).
Формат ввода: Название [УНН], ФИО владельца, Дата основания

МАРШРУТ:
номер маршрута (целое четырехзначное число);
станция отправления (строка 15 символов);
конечная станция (строка 15 символов);
время отправления (в формате чч:мм);
время прибытия (в формате чч:мм).
Формат ввода: Номер маршрута Станция отправления (Время отправления) – Конечная станция (Время прибытия)

ТЕЛЕФОН:
номер (целое семизначное число, уникальное поле);
ФИО владельца (строка 30 символов);
дата подключения (в формате дд.мм.гггг);
тарифный план (строка 15 символов).
Формат ввода: Номер ФИО владельца Дата подключения «Тарифный план»

ВИДЕОДИСК:
серийный номер (целое десятизначное число)
название (строка 20 символов);
продолжительность (целое число, в минутах);
кинокомпания (строка 15 символов);
жанр (0 – боевик, 1 – комедия, 2 – драма и т.д.).
Формат ввода: Серийный номер «Название фильма» Кинокомпания (Жанр), Продолжительность

ПОЕЗД:
номер (целое трехзначное число)
название (строка 20 символов);
станция отравления (строка 20 символов);
станция назначения (строка 20 символов);
количество вагонов (целое число меньше 30).
Формат ввода: Номер «Название» (Станция отправления – Станция назначения), Количество вагонов

УСТРОЙСТВО:
тип устройства (строка 15 символов);
модель (строка 10 символов);
фирма производитель (строка 15 символов);
серийный номер (цифро-символьный код длиной 20 знаков);
дата изготовления (в формате дд.мм.гггг).
Формат ввода: Тип устройства Модель (Фирма), Номер, Дата изготовления

Таблица 1.7.2 – Индивидуальные варианты к заданиям №2,3

№	Задание на разработку классов	Метод обобщение
1	Матрица, вектор	Сложение
2	Матрица, дробь	вычитание
3	Матрица, комплексное число	Умножение на объект
4	Матрица, рациональное число	Умножение на число
5	Полином, вектор	Сложение
6	Полином, дробь	вычитание
7	Полином, комплексное число	Умножение на объект
8	Полином, рациональное число	Умножение на число
9	Дробь, вектор	Сложение
10	Комплексное число, рациональное число	вычитание
11	Вектор, комплексное число	Умножение на объект
12	Вектор, рациональное число	Умножение на число

13	Дробь, комплексное число	Сложение
14	Дробь, рациональное число	вычитание
15	Матрица, полином	Умножение на объект