Урок 3

Методы. От структур к объектам

Подробнее поговорим о методах. От структур перейдем к объектам.

Методы

- Модификатор параметров методов
  - Задача 1. Написать функцию обмена значениями двух переменных
  - Задача 2. Организовать ввод с «защитой от дурака»

Структуры

Структуры для работы со временем

От структур к объектам

- Примеры
  - Класс для работы с комплексными числами. Вариант 1
  - Класс для работы с комплексными числами. Вариант 2
- Статические поля и методы

Класс для генерации случайных чисел

Практическая часть урока

- Задача 1. Найти максимальное число
- Задача 2. Вычислить частное q и остаток r при делении a на d, не используя операций деления (/) и взятия остатка от деления (%)

Задача 3. Написать программу табуляции произвольной функции в диапазоне от a до b

б) Решение с использованием ООП

Задача 4. Игра «Угадай число»

Структура и класс Point

Класс Point

Домашнее задание

Дополнительные материалы

Используемая литература

Методы

Метод — это блок кода, содержащий ряд инструкций. Программа инициирует выполнение инструкций, вызывая метод, указывая все значения аргументов, необходимые для этого метода.

Модификатор параметров методов

Задача. Написать метод, внутри которого поменять знаки нескольким переменным.

```
static void ChangeSign(ref int a, ref int b, ref int c)
{
      a = -a;
      b = -b;
      c = -c;
}
```

По умолчанию внутрь методов передаются копии переменных. Это называется передача по значению. Так сделано, чтобы обезопасить переменные основной программы. Если же мы хотим внутри подпрограммы изменить значения переменных, требуется использовать модификаторы.

Передача по ссылке — ref. В этом случае будет передаваться ссылка на переменную, внутри метода переменную можно будет изменить. Но так как в .NET Framework для обеспечения безопасного программирования нельзя использовать переменные, которым не присвоены начальные значения, значения придется присваивать.

Передача по ссылке без обязательной первоначальной инициализации — out. При этом внутри метода переменной обязательно должно быть присвоено значение.

Задача 1. Написать функцию обмена значениями двух переменных.

```
using System;

    class Program
    {
        static void Swap(ref int a, ref int b)
        {
            int t = a;
            a = b;
            b = t;
        }
        static void Main(string[] args)
        {
            int a = 10;
            int b = 20;
            Console.WriteLine("a={0} b={1}", a, b);
            Swap(ref a, ref b);// Пример вызова
            Console.WriteLine("a={0} b={1}", a, b);
        }
    }
```

Задача 2. Организовать ввод с «защитой от дурака».

Для ввода данных с проверкой можно использовать метод TryParse. TryParse передаёт результат работы через параметр метода и возвращает информацию, правильно или неправильно произошёл перевод.

```
using System;

    class Program
    {
        static int value;
        static string console_message = "Введите число:";
        static int GetValue(string message)
        {
            int x;
            string s;
               bool flag;          // Логическая переменная, выступающая в роли
"флага".
                                   // Истинно (флаг поднят), ложно (флаг опущен)
            do
            {
                Console.WriteLine(message);
                s = Console.ReadLine();
                                        //  Если перевод произошел неправильно, то
результатом будет false
                flag = int.TryParse(s, out x);
            }
            while (!flag);  //  Пока false(!false=true), повторять цикл
            return x;
        }
        static void ShowValue(string description)
        {
            Console.WriteLine(description + value);
        }
        static int ReturnValue()
        {
            ShowValue("ReturnValue (до): ");
            int tmp = 10;
            ShowValue("ReturnValue (после): ");
            return tmp;
        }
        static void OutParameter(out int tmp)
        {
            ShowValue("OutParameter (до): ");
            tmp = 10;
            ShowValue("OutParameter (после): ");
        }
        static void Main()
        {
            value = GetValue(console_message);
            Console.WriteLine("Return ...");
            value = ReturnValue();
            ShowValue("value после ReturnValue(): ");

            value = GetValue(console_message);
            Console.WriteLine("Out parameter ...");
            OutParameter(out value);
```

```
            ShowValue("value после OutParameter(): ");
        }
    }
```

Модификатор out не требует обязательной начальной инициализации переменной, но в методе обязательно должно быть присвоено параметру значение.

Структуры

Кроме базовых элементарных типов данных и перечислений, в C# имеется и составной тип данных, который называется структурой. Структуры могут содержать в себе обычные переменные и методы.

Пример структуры в C#, представляющей комплексное число и несколько действий с ним:

```
using System;

    struct Complex
    {
        public double im;
        public double re;
        //  в C# в структурах могут храниться также действия над данными
        public Complex Plus(Complex x)
        {
            Complex y;
            y.im = im + x.im;
            y.re = re + x.re;
            return y;
        }
        //  Пример произведения двух комплексных чисел
        public Complex Multi(Complex x)
        {
            Complex y;
            y.im = re * x.im + im * x.re;
            y.re = re * x.re - im * x.im;
            return y;
        }
        public string ToString()
        {
            return re + "+" + im + "i";
        }
    }
    class Program
    {
        static void Main(string[] args)
        {
            Complex complex1;
            complex1.re = 1;
            complex1.im = 1;

            Complex complex2;
            complex2.re = 2;
            complex2.im = 2;

            Complex result = complex1.Plus(complex2);
```

```
        Console.WriteLine(result.ToString());
            result = complex1.Multi(complex2);
            Console.WriteLine(result.ToString());
        }
    }
```

Структуры относятся к типу *значения*. В примере при присвоении переменной *b* переменной *a* в *b* переносятся значения полей *a*.

```
Complex a,b;
a.im=1;
a.re=2;
b=a;      // В поля структуры b скопируется a
```

Структуры для работы со временем

В .NET Framework огромное количество уже готовых структур. Вот некоторые из них:

1. DateTime хранит в себе дату и время.
2. TimeSpan предназначен для хранения промежутков времени.

Особенно полезно знать, что при вычитании DateTime получается структура TimeSpan.

```
using System;
using System.Threading;

    class Program
    {
        static void Main(string[] args)
        {
            DateTime start = DateTime.Now;
            Console.WriteLine(start);
            Thread.Sleep(2000);
            DateTime finish = DateTime.Now;
            Console.WriteLine(finish);
            TimeSpan duration = finish - start;
            Console.WriteLine(duration);
        }
    }
```

От структур к объектам

Чтобы объяснить, что такое объект, нужно понять, что такое класс. Чтобы разобрать понятие класса, проведём аналогию со структурой.

Когда мы описывали структуру, мы как бы описывали будущую переменную, не создавая ее. Когда же мы описываем переменную типа структуры, мы выделяем место в памяти для структуры — появляется конкретный экземпляр структуры. Так же и с классом.

Класс — это будущий объект. В классе, как и в структуре, мы описываем будущий объект. Для создания объекта нужно сначала объявить переменную — ссылку на объект, а потом попросить выделить место в памяти для объекта.

Описание класса:

```
class MyClass                                    // Описали пустой класс
{
}

MyClass myObj=new MyClass();    // Создали объект класса MyClass
```

Объекты относятся к ссылочным типам. Это означает, что их нужно создавать, используя слово new. Для объектов выделяется место в куче (heap), и переменная хранит ссылку на это место. При присвоении объекта другому объекту переносится адрес ссылки. То есть две переменные указывают на одну область памяти, или, другими словами, две переменные указывают на один и тот же объект.

```
Object a=new Object();          // Создали новый объект класса Object
Object b;
b=a;                            // В b хранится ссылка на объект a
```

Почему же тогда нужны классы и структуры, если получается, что это почти одно и то же? Дело в том, что структуры были придуманы еще до появления ООП, их назначение было просто объединить в одном типе несколько данных. Структуры не поддерживают один из основных принципов ООП — наследование.

Структуру стоит использовать, когда она хранит в себе небольшой объём информации.

Примеры

Класс для работы с комплексными числами. Вариант 1.

```
using System;

    class Complex
    {
            // Все методы и поля публичные. Мы можем получить доступ к ним из
другого класса.
        public double im;
        public double re;

        public Complex Plus(Complex x2)
        {
            Complex x3 = new Complex();
            x3.im = x2.im + this.im;
            x3.re = x2.re + this.re;
            return x3;
        }
```

```
        public string ToString()
        {
            return re + "+" + im + "i";
        }
    }

    class Program
    {
        static void Main(string[] args)
        {
            Complex complex1 = new Complex();
            complex1.re = 1;
            complex1.im = 1;

            Complex complex2 = new Complex();
            complex2.re = 2;
            complex2.im = 2;

            Complex result = complex1.Plus(complex2);
            Console.WriteLine(result.ToString());
        }
    }
```

Правда, в этом случае мы нарушаем один из принципов ООП об инкапсуляции данных. Это означает, что данные объекта должны быть скрыты от прямого изменения. Но если их скрыть, как же получить к ним доступ? Для этого существуют различные механизмы. Рассмотрим два из них: использование конструктора при создании объекта и механизм свойств.

Класс для работы с комплексными числами. Вариант 2.

```
using System;

class Complex
{
    // Поля приватные.
    private double im;
    // По умолчанию элементы приватные, поэтому private можно не писать.
    double re;

    // Конструктор без параметров.
    public Complex()
    {
        im = 0;
        re = 0;
    }

    // Конструктор, в котором задаем поля.
    // Специально создадим параметр re, совпадающий с именем поля в классе.
    public Complex(double _im, double re)
    {
   // Здесь имена не совпадают, и компилятор легко понимает, что чем является.
        im = _im;
        // Чтобы показать, что к полю нашего класса присваивается параметр,
        // используется ключевое слово this
```

```
        // Поле параметр
        this.re = re;
    }
    public Complex Plus(Complex x2)
    {
        Complex x3 = new Complex();
        x3.im = x2.im + im;
        x3.re = x2.re + re;
        return x3;
    }
    // Свойства - это механизм доступа к данным класса.
    public double Im
    {
        get { return im; }
        set
        {
            // Для примера ограничимся только положительными числами.
            if (value >= 0) im = value;
        }
    }
    // Специальный метод, который возвращает строковое представление данных.
    public string ToString()
    {
        return re + "+" + im + "i";
    }
}
class Program
{
    static void Main(string[] args)
    {
        // Описали ссылку на объект.
        Complex complex1;
        // Создали объект и сохранили ссылку на него в complex1.
        complex1 = new Complex(1, 1);
        // Описали объект и создали его.
        Complex complex2 = new Complex(2, 2);
        // С помощью свойства Im изменили внутреннее (приватное) поле im.
        complex2.Im = 3;
        // Создали ссылку на объект.
        Complex result;
        // Так как в методе Plus создается новый объект,
        // в result сохраняем ссылку на вновь созданный объект.
        result = complex1.Plus(complex2);
        Console.WriteLine(result.ToString());
    }
}
```

Статические поля и методы

Давайте разберёмся, что означает слово static в начале метода Main.

Пример:

```
using System;
    class MyClass
```

```
    {
        public static int static_a;
        public int non_static_a;
    }
    class Program
    {
        static void Main(string[] args)
        {
            MyClass.static_a = 10;
            MyClass myobj = new MyClass();
            myobj.non_static_a = 10;
        }
    }
```

Статическое поле принадлежит всем объектам класса. Или, другими словами, статическое поле принадлежит классу. Для доступа к нестатическому полю требуется создание объекта класса. Метод Main и другие, созданные до этого, были статическими, чтобы можно было обращаться к методу без создания объекта.

То, что мы узнали об ООП, — это лишь вершина айсберга, но этого достаточно, чтобы начать писать программы, используя принципы ООП. В дальнейшем изучении ООП нам помогут массивы.

Класс для генерации случайных чисел

Далее познакомимся с массивами. Так как массив предназначен для хранения нескольких элементов, чтобы не заполнять массив вручную, полезно познакомиться с классом Random. Класс Random содержит в себе несколько методов для генерации случайных чисел, самый полезный из которых — Next.

Пример генерации последовательности из 10 случайных чисел в диапазоне от 0 до 10:

```
using System;

    class Program
    {
        static void Main(string[] args)
        {
            Random rnd = new Random();
              for (int i = 0; i < 10; i++) Console.Write("[{0,5}]", rnd.Next(0,
10));
            Console.WriteLine();
        }
    }
```

Практическая часть урока

Задача 1. Найти максимальное число.

На вход программе подаётся последовательность чисел, заканчивающаяся нулём. Найти максимальное число.

```
using System;


    class Program
    {
        static void Main()
        {
            int a = int.Parse(Console.ReadLine());
            int max = a;
            while (a != 0)
            {
                a = int.Parse(Console.ReadLine());
                if (a > max) max = a;
            }
            Console.WriteLine(max);
        }
    }
```

Для закрепления выполните считывание данных из файла.

Задача 2. Вычислить частное q и остаток r при делении a на d, не используя операций деления (/) и взятия остатка от деления (%).

Дано натуральное (целое неотрицательное) число a и целое положительное число d. Вычислить частное q и остаток r при делении a на d, не используя операций деления (/) и взятия остатка от деления (%).

```
using System;

    class Program
    {
        static void Main()
        {
            int a = 10;
            int d = 3;
            // a/d
            int r = a, q = 0;
            while (r >= d)
            {
                r = r - d;
                q++;
            }
            Console.WriteLine("Частное {0}.\n Остаток {1}", q, r);
        }
    }
```

Задача 3. Написать программу табуляции произвольной функции в диапазоне от a до b.

Написать программу табуляции произвольной функции в диапазоне от a до b. Функция задаётся программно.

а) Решение без использования ООП.

```
using System;
class Program
{
    static double F(double x)
    {
        return 1/x;
    }
    static void Main()
    {
        double a = -5;
        double b = 5;
        double h = 0.5;
        Console.WriteLine("{0,10}{1,10}", "x", "F(x)");
        for (double x= a;x<= b;x=x+h)
        {
            Console.WriteLine("{0,10}{1,10:f3}", x,F(x));
        }
    }
}
```

Попробуйте переделать программу, используя циклы while и do while. Реализуйте вывод в файл.

б) Решение с использованием ООП.

```
using System;
class Table
{
    double a = -5;
    double b = 5;
    double h = 0.5;

    public Table()
    {
    }
    public Table(double a, double b,double h)
    {
        this.a = a;
        this.b = b;
        this.h = h;
    }

    double F(double x)
    {
        return 1 / x;
    }

    public void Show(double a, double b, double h)
    {
        Console.WriteLine("{0,10}{1,10}", "x", "F(x)");
        for (double x = a; x <= b; x = x + h)
        {
            Console.WriteLine("{0,10}{1,10:f3}", x, F(x));
        }
    }
```

```
    public void Show()
    {
        Console.WriteLine("{0,10}{1,10}", "x", "F(x)");
        for (double x = a; x <= b; x = x + h)
        {
            Console.WriteLine("{0,10}{1,10:f3}", x, F(x));
        }
    }
}
class Program
{
    static void Main()
    {
        Table table1 = new Table();
        table1.Show();
        Console.WriteLine("Для выполнения следующего расчета нажмите любую
клавишу");
        Console.ReadKey();
        Table table2 = new Table(1, 2, 0.5);
        table2.Show();
        Console.ReadKey();
    }
}
```

Задача 4. Игра «Угадай число».

Игра «Угадай число». Метод половинного деления.

Написать игру «Угадай число». Компьютер загадывает число в диапазоне от 1 до 100, а человек за ограниченное число попыток должен угадать его. Количество попыток вычисляется по формуле $i=\log_2 N+1$

```
using System;
    class Program
    {
        static void Main(string[] args)
        {
            int min = 1;
            int max = 100;
            int maxCount = (int)Math.Log(max - min + 1, 2)+1;
            int count = 0;
            Random rnd = new Random();
            int guessNumber = rnd.Next(min, max);
                  Console.WriteLine("Компьютер загадал число от {0} до {1}.
Попробуйте угадать его за {2} попыток", min, max, maxCount);
            int n;
            do
            {
                count++;
                Console.Write("{0} попытка. Введите число:", count);
                n = int.Parse(Console.ReadLine());
                if (n > guessNumber) Console.WriteLine("Перелет!");
                if (n < guessNumber) Console.WriteLine("Недолет!");
            }
            while (count < maxCount && n!=guessNumber);
               if (n == guessNumber) Console.WriteLine("Поздравляю! Вы угадали
число за {0} попыток", count);
```

```
            else Console.WriteLine("Неудача. Попробуйте еще раз");
        }
    }
```

Структура и класс Point

```
using System;

    class Program
    {
        struct Point
        {
            double _x, _y;

            public Point(double x, double y)
            {
                _x = x;
                _y = y;
            }

            public double Distance(Point Z)
            {
                  return Math.Sqrt(Math.Pow(_x - Z._x, 2) + Math.Pow(_y - Z._y,
2));
            }
        }

        static void Main(string[] args)
        {
            Point C;

            Point A = new Point(0, 0);
            Point B = new Point(2, 2);
            Console.WriteLine(A.Distance(B));
        }
    }
```

Класс Point

```
using System;

    class Program
    {
        class Point
        {
            double _x, _y;

            public Point()
            {
                _x = _y = 0;
            }
            public Point(double x, double y)
```

```
            {
                _x = x;
                _y = y;
            }
            public void SetX(double value)
            {
                _x = value;
            }
            public double GetX()
            {
                return _x;
            }
            public double X
            {
                get { return _x; }
                set
                {
                    if (value > 0) _x = value;
                    else _x = -value;
                }
            }
            public double Distance(Point Z)
            {
                 return Math.Sqrt(Math.Pow(_x - Z._x, 2) + Math.Pow(_y - Z._y,
2));
            }
            public override string ToString()
            {
                return _x + "," + _y;
            }
        }

        static void Main(string[] args)
        {
            Point C = new Point();
            C.SetX(10);
            Console.WriteLine(C.GetX());
            Console.WriteLine(C.X);
            C.X = 10;
            Console.WriteLine(C);

            Point A = new Point();
            Point B = new Point(2, 2);
            Console.WriteLine(A.Distance(B));
            Console.ReadKey();
        }
    }
```

Домашнее задание

1. а) Дописать структуру Complex, добавив метод вычитания комплексных чисел. Продемонстрировать работу структуры.
   б) Дописать класс Complex, добавив методы вычитания и произведения чисел. Проверить работу класса.
   в) Добавить диалог с использованием switch демонстрирующий работу класса.

2. а) С клавиатуры вводятся числа, пока не будет введён 0 (каждое число в новой строке). Требуется подсчитать сумму всех нечётных положительных чисел. Сами числа и сумму вывести на экран, используя tryParse.
3. *Описать класс дробей — рациональных чисел, являющихся отношением двух целых чисел. Предусмотреть методы сложения, вычитания, умножения и деления дробей. Написать программу, демонстрирующую все разработанные элементы класса.
   * Добавить свойства типа int для доступа к числителю и знаменателю;
   * Добавить свойство типа double только на чтение, чтобы получить десятичную дробь числа;
   ** Добавить проверку, чтобы знаменатель не равнялся 0. Выбрасывать исключение ArgumentException("Знаменатель не может быть равен 0");
   *** Добавить упрощение дробей.

Достаточно решить 2 задачи. Все программы сделать в одном решении.

Дополнительные материалы

1. Павловская Т.А. Программирование на языке высокого уровня. Глава «Классы: основные понятия». — СПб: Питер, 2009.
2. Троелсен Э. Язык программирования C# 5.0 и платформа .NET 4.5. Глава 4 «Типы структур». — М.: ООО «И.Д. Вильямс», 2013.
3. Троелсен Э. Язык программирования C# 5.0 и платформа .NET 4.5. Глава 4 «Типы значений и ссылочные типы». — М.: ООО «И.Д. Вильямс», 2013.

Используемая литература

Для подготовки данного методического пособия были использованы следующие ресурсы:

1. Павловская Т.А. Программирование на языке высокого уровня. — СПб: Питер, 2009.
2. Троелсен Э. Язык программирования C# 5.0 и платформа .NET 4.5. — М.: ООО «И.Д. Вильямс», 2013.
3. Шилдт Г. C# 4.0. Полное руководство. — М.: ООО «И.Д. Вильямс», 2011.
4. MSDN.