МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РФ

Брянский государственный технический университет

Утверждаю
Ректор университета
_____О.Н. Федонин
«_____»_____2013 г.

ЯЗЫКИ ПРОГРАММИРОВАНИЯ

РАБОТА С ПОЛЬЗОВАТЕЛЬСКИМИ МЕТОДАМИ КЛАССА

Методические указания
к выполнению лабораторной работы
для студентов очной формы обучения
специальностей 090303 – «Информационная безопасность автоматизированных систем»,
090900 – «Информационная безопасность»

Брянск 2013

УДК 004.43

Языки программирования. Работа с пользовательскими методами класса: методические указания к выполнению лабораторной работы для студентов очной формы обучения специальностей 090303 – «Информационная безопасность автоматизированных систем», 090900 – «Информационная безопасность». – Брянск: БГТУ, 2013. – 12 с.

Разработали:
Ю.А. Леонов, к.т.н., доц.,
Е.А. Леонов, к.т.н., доц.

Научный редактор: Ю.М. Казаков
Редактор издательства: Л.И. Афонина
Компьютерный набор: Ю.А. Леонов

Рекомендовано кафедрой «Компьютерные технологии и системы» БГТУ (протокол № 2 от 19.09.2013)

Темплан 2013 г., п.

Подписано в печать Формат 60x84 1/16. Бумага офсетная.
Офсетная печать.
Усл. печ. л. 0,7 Уч. – изд. л. 0,7 Тираж 20 экз. Заказ Бесплатно

Издательство брянского государственного технического университета,
241035, Брянск, бульвар 50-летия Октября, 7, БГТУ. 58-82-49
Лаборатория оперативной полиграфии БГТУ, ул. Харьковская, 9

1. ЦЕЛЬ РАБОТЫ

Целью работы является изучение общей структуры методов класса, а также овладение практическими навыками их использования при решении задач.

Продолжительность работы – 4 ч.

2. ТЕОРЕТИЧЕСКАЯ ЧАСТЬ

При создании структур и классов можно описывать функции, которые предназначены для выполнения некоторого алгоритма. В объектной модели программирования эти функции называются методами.

Методом называется член (элемент) класса или структуры, в котором описываются конструкции, предназначенные для выполнения определенного алгоритма.

Методы в общей структуре класса

class <имя класса> {
    [<поля>]
    [<константы>]
    [<свойства>]
    [<методы>]
    [<события>]
    [<перегруженные операторы>]
    [<индексаторы>]
    [<конструкторы>]
    [<деструкторы>]
    [<интерфейсы>]
    [<делегаты>]
    [<типы данных>]
}

Как видно методы описываются внутри класса, они должны быть описаны в соответствии с установленным синтаксисом описания.

Синтаксис описания метода

[<модификатор доступа>] [static] <тип возвращаемого значения> | void <имя метода> ([<формальные параметры метода>]) {<операторы> [return <возвращаемое значение>]}

Для членов структуры и класса можно устанавливать модификаторы доступа, в частности для методов класса.

Существуют следующие модификаторы доступа:

private – метод доступен только из текущего класса (применяется по умолчанию);

public – метод доступен из любой точки программного кода в текущем пространстве имен;

protected – метод доступен из текущего класса (как private), а также из любого производного класса.

internal – метод доступен из любого кода в той же сборке, но не из другой сборки.

protected internal – метод доступен из любого кода в сборке, в которой он объявлен, или из наследованного класса другой сборки.

Ключевое слово «static» сообщает компилятору о том, что необходимо метод разместить в статической области памяти, которая на время выполнения программы не может быть занята чем-то другим. Таким образом, статический метод доступен напрямую из класса до создания экземпляров класса (объектов).

Тип возвращаемого значения может быть как встроенным, так и пользовательским.

Формальные параметры перечисляются через запятую с указанием типа данных и имени параметра.

Для примера опишем метод, который возвращает значение функции квадратного уравнения: $f(x) = ax^2 + bx + c$.

```
private double Quadratic(double a, double b, double c,
double x)
{
    return a * x * x + b * x + c;
}
```

Если метод не возвращает данные в своем имени, то вместо типа возвращаемого значения указывается слово «void» (в переводе с англ. «пустой»).

Например:

```
private void PrintMsg(string msg)
{
    Console.WriteLine(msg);
}
```

В случае если параметры метода отсутствуют, круглые скобки всё равно ставить необходимо:

```
private void PrintMsg()
{
    Console.WriteLine("Текст сообщения");
}
```

После того как метод описан его можно вызвать.

Синтаксис вызова метода

<имя метода>([<список фактических параметров>]);

Если метод вызывается не из текущего класса, то перед методом через точку «.» должен быть указан объект, в котором находится метод или если метод статический, то класс.

Синтаксис вызова нестатического метода

<имя объекта>.<имя метода> ([<список фактических параметров >]);

Синтаксис вызова статического метода

<имя класса>.<имя метода> ([<список фактических параметров>]);

Для описания формальных параметров существуют различные модификаторы:

(модификатор отсутствует) – входной параметр (по умолчанию); при входе в метод выделяется память под формальный параметр и копируется туда пришедшее значение из фактического параметра;

ref – передается ссылка (reference), т.е. у фактического и формального параметров один адрес, таким образом, при присваивании значения любому из параметров (соответствующим друг другу формальному или фактическому) во втором параметре также «видно» это значение;

out – исходящий (выходной) параметр, который передается в соответствующий фактический параметр;

params – позволяет передавать набор параметров как единое целое (всегда последний в списке формальных параметров).

При использовании модификаторов *ref* и *out* необходимо их указывать не только при описании формальных параметров, но и при описании фактических параметров.

Рассмотрим примеры использования описанных модификаторов формальных параметров.

Пример использования *входного параметра*:

```
class Program
{
    static void Method(int p1)
    {
        Console.Write(p1);
        p1 = 3;
    }
    static void Main()
    {
        int a = 5;
        Method(a);
        Console.Write(a);
        Console.Read();
    }
}
```

Результат на экране: 55.

При входе в метод выделяется новая память под параметр *p*1 и туда копируется значение из переменной *a*. Несмотря на изменение формального параметра *p*1 фактический параметр *a* не изменился, т.к. эти параметры не связаны.

Пример использования модификатора «*out*»

```
class Program
{
    static void Method(out int p1)
    {
        p1 = 3;
        Console.Write(p1);
    }
    static void Main()
    {
        int a = 5;
        Method(out a);
        Console.Write(a);
        Console.Read();
```

```
    }
}
```

Результат на экране: 33.

Перед вызовом метода *Method* фактическому параметру было присвоено значение 5, но внутри этого метода формальному параметру *p*1 было присвоено новое значение – 3. Так как этот параметр описан с модификатором *out*, произошел возврат нового значения через формальный параметр *p*1 в фактический параметр *a*. Таким образом, при выходе из метода переменная *a* имеет новое значение – 3.

Пример использования модификатора «*ref*»

```
class Program
{
    static void Method(ref int p1)
    {
        Console.Write(p1);
        p1 = 3;
    }
    static void Main()
    {
        int a = 5;
        Method(ref a);
        Console.Write(a);
        Console.Read();
    }
}
```

Результат на экране: 53.

Формальный параметр *p*1 и фактический параметр *a* связаны одним адресом. При изменении любого из параметров автоматически будет изменяться и другой параметр.

Пример использования модификатора «*params*»

```
class Program
{
    private static void Method(params int[] array)
    {
        foreach (int elem in array)
            Console.Write("{0, 3}", elem);
        Console.WriteLine();
    }
    static void Main()
```

```
    {
        Method(10);
        Method(10, 20, 30);
        int[] array = new int[5] { 1, 2, 3, 4, 5 };
        Method(array);
        Console.Read();
    }
}
```

Результат на экране:
10
10 20 30
1 2 3 4 5.

Если заранее неизвестно какое количество параметров будет передано, то можно использовать модификатор «*params*». В этом случае тип формального параметра будет одномерным массивом. При таком объявлении возможна передача различного количества фактических параметров, а также можно передавать одномерный массив.

3. ПОРЯДОК ВЫПОЛНЕНИЯ РАБОТЫ

Для выполнения лабораторной работы необходимо написать программу на языке C#, которая будет демонстрировать работу созданных методов.

В работе необходимо описать два метода, которые реализуют один и тот же алгоритм, описанный в списке заданий. Первый метод для возврата результата должен использовать имя метода, а второй должен возвращать результат через параметры.

Обязательные требования, предъявляемые к созданию методов.

1. Входные данные необходимо передавать через параметры.
2. Названия переменных, констант и методов должны быть логически обоснованы и давать понятие о том, что в них предполагается хранить или обрабатывать.
3. Программа должна запрашивать входные данные и выводить итоговый результат с пояснениями.

Пример выполнения задания

Задача: Необходимо написать программу на языке C#, которая будет искать цифры в передаваемой строковой переменной.

Нахождение цифр требуется реализовать с помощью двух методов. Первый метод должен возвращать результат в имени функции, а второй через параметры. В случае если хотя бы одна цифра имеется в строке, то метод должен возвращать значение *true* или *false* в противном случае.

Решение (программный код на языке C#)

```
class Program
{
    static void FindDigit(string s, out bool digit)
    {
        for (int i = 0; i < s.Length; i++)
            if (char.IsDigit(s[i]))
            {
                digit = true;
                return;
            }
        digit = false;
    }
    static bool FindDigit(string s)
    {
        for (int i = 0; i < s.Length; i++)
            if (char.IsDigit(s[i])) return true;
        return false;
    }

    static void Main()
    {
        Console.WriteLine("Введите строку");
        string s = Console.ReadLine();
        bool digit;
        FindDigit(s, out digit);
        if (digit) Console.WriteLine("В данной строке
        имеются цифры (возвращение результата через
        параметры).");
        else Console.WriteLine("В данной строке цифр нет
        (возвращение результата через параметры).");
        if (FindDigit(s)) Console.WriteLine("В данной
        строке имеются цифры (возвращение результата через
        имя метода).");
        else Console.WriteLine("В данной строке цифр нет
        (возвращение результата через имя метода).");
```

```
            Console.Read();
        }
    }
```

4. СПИСОК ЗАДАНИЙ

Таблица 1

№ варианта	Задание
	Необходимо создать метод, который:
1	удаляет из строки наибольшее и наименьшее числа;
2	“вырезает” из строковой переменной цифры;
3	“вырезает” из строковой переменной буквы;
4	считает сумму нечетных чисел, находящихся в строке;
5	проводит зашифровывание (расшифровывание) переданного слова кодом Цезаря с ключом *n*;
6	определяет количество заданных букв в строковой переменной; буква и строковая переменная передаются через фактические параметры;
7	решает квадратное уравнение; коэффициенты *a*, *b*, *c* задаются во входных параметрах;
8	проверяет корректность введенной даты; формат даты передается в строковой переменной, где *d* – день, *m* – месяц, *y* – год; количество этих букв говорит о количестве позиций дня, месяца или года;
9	подсчитывает сумму только нечетных чисел одномерного массива, которые находятся на четных позициях;
10	подсчитывает количество дней до Нового года относительно текущей даты;
11	вычисляет сумму капитала, положенного в банк под определенный процент; сумма вложения и процентная ставка, а также срок вложения задаются во входных параметрах; каждый год сумма, с которой начисляют проценты, меняется в зависимости от текущего капитала;
12	выводит индекс строки двумерного массива размером *n***m* с наибольшим средним арифметическим значением;
13	определяет знак зодиака человека по введенной дате рождения;
14	вычисляет необходимое время для передачи файла; известна скорость сети в Кбит/с, размер файла, который необходимо пере-править по сети, а также известно, что каждый четвертый пакет размером в 1 байт теряется в сети;
15	проверяет можно ли составить переданное слово из букв русского алфавита, содержащихся в строке;

Окончание табл. 1

16	переставляет в обратном порядке элементы одномерного массива, расположенные между максимальным и минимальным элементами;
17	определяет, возможен ли такой треугольник, если даны размеры трех сторон треугольника;
18	выводит уравнение прямой в строковом типе, если известны координаты двух точек;
19	строит таблицу умножения; вывод таблицы производится в виде нескольких столбцов;
20	суммирует элементы главной и побочной диагоналей двумерного массива размера *n***n* и выводит наибольшую сумму;
21	определяет, сколько раз встречается заданная буква в слове, и выводит порядковые номера буквы в слове;
22	определяет возраст человека на данный момент времени по введенной дате рождения;
23	выводит информацию о значении строковой переменной: количество гласных, согласных букв, количество цифр;
24	переводит декартовы координаты в полярные;
25	переводит полярные координаты в декартовы;
26	вычисляет факториал заданного числа;
27	вычисляет количество слогов в слове;
28	определяет процент в рублях от задаваемой суммы; процент и сумма задаются во входных параметрах;
29	вычисляет определитель матрицы 2-го порядка;
30	переводит логические координаты в физические; соотношение их между собой задается константой.

5. КОНТРОЛЬНЫЕ ВОПРОСЫ

1. В каком месте программы описываются методы?
2. В чем различие статических методов от нестатических?
3. Какие модификаторы доступа к членам класса и структуры вы знаете?
4. Какие модификаторы для описания формальных параметров вы знаете? Для чего предназначен каждый из них?
5. В чем отличие модификаторов *ref* и *out*?
6. Какова структура описания метода?
7. Опишите синтаксис вызова метода.
8. Как описать метод, который не возвращает никаких данных в своем имени?

9. Что такое область видимости членов класса или структуры?
10. Какие параметры называются формальными, а какие фактическими?

6. СПИСОК РЕКОМЕНДУЕМОЙ ЛИТЕРАТУРЫ

Основная

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.

Дополнительная

4. Джесс Либерти. Программирование на C#. – Изд.: КноРус, 2003. – 688с.
5. Харви Дейтел. C# в подлиннике. Наиболее полное руководство. – Изд.: БХВ-Петербург, 2006. – 1056с.