**Лабораторная работа № 7. Разработка алгоритмов и программ**

**по решению задач с использованием методов**

**1 Цель работы**

Изучить операторы, используемые для организации подпрограмм (методов).

**2 Порядок выполнения работы**

Получить задание для выполнения лабораторной работы (раздел 5) согласно своему варианту (номер студента по журналу). Разработать и отладить программу. Составить отчет о лабораторной работе и защитить его у преподавателя.

**3 Содержание отчета**

* наименование и цель работы;
* задание на лабораторную работу согласно варианту;
* схема алгоритма, текст программы на алгоритмическом языке;
* результаты работы программы.

**4 Краткая теория**

Метод – именованный блок кода, решающий единственную задачу и выделенный с целью переиспользования. Метод определяется на уровне класса и в общем случае имеет следующую структуру:

<модификаторы> типВозвращаемогоЗначения названиеМетода (<параметры>)

{

// тело метода

}

У каждого метода обязательно должен быть определен модификатор метода, название и тело.

На самом деле, Console.WriteLine Console.ReadLine также является примером методов, поэтому примерное взаимодействие с ними уже знакомо. В коде ниже представлен пример определения метода:

using System;

using System.Collections.Generic;

using System.Data;

using System.IO;

using System.Linq;

using System.Reflection.Emit;

using System.Runtime.CompilerServices;

using System.Text;

using System.Threading;

using System.Threading.Tasks;

using System.Xml.Linq;

namespace ExampleApp

{

internal class Program

{

static void Main(string[] args)

{

Greetings("Aleksey");

}

static void Greetings(string name)

{

Console.WriteLine($"Hello, {name}");

}

}

}

Про ключевое слово static более подробно будет рассказано в контексте объектно-ориентированного программирования, поэтому пока пропустим его описание. Следующее ключевое слово, void, указывает, что метод не возвращает никакого значения в возвращаемый код, а просто выполняет некоторые действия. На третьем месте идет название метода, Greetings. Его разработчик придумывает сам, но как практика хорошего кода, название метода должно начинаться с заглавной буквы. Последним идут параметры метода, они передаются в круглых скобках. Для каждого параметра сначала указывается используемый тип, а потом название параметра, которое будет доступно только в этом методе. Это значит, что к переменной name можно обратится только внутри метода Greetings, в методе main такой переменной просто не будет существовать.

Для того, чтобы вызвать метод, необходимо обратится к его имени и передать необходимые параметры. В примере, в качестве параметра name передается значение “Aleksey”. Параметры должны соответствовать при передаче по типу данных. Это значит, что если методу необходима строка, то не получится в нее передать целое число без преобразования. В случае попытки передачи неподходящего аргумента компилятор сообщит об ошибке.

Также, можно воспользоваться необязательными параметрами. Для этого при определении метода для параметра необходимо определить значение по умолчанию. Для демонстрации их работы необходимо изменить заголовок метода на следующую строку:

static void Greetings(string name = “Sergey”)

При этом, вызов метода в методе Main можно сократить до:

Greetings();

Очень часто методы могут содержать несколько параметров и разработчик имеет право передавать их внутрь метода в том порядке, в котором ему будет удобно. Однако, для этого необходимо воспользоваться именованными параметрами при вызове. Они доступны и для метода Greetings. Вызов этого метода с именованными параметрами будет выглядеть следующим образом:

Greetings(name: “Aleksey”);

Теперь необходимо рассмотреть случай, когда метод возвращает какое-то значение. Примеры таких методов были разобраны при рассмотрении методов библиотеки Math. Чтобы организовать такие методы, на месте ключевого слова void должен быть указан тот тип, который будет возвращать метод, а само тело метода должно оканчиваться оператором return после которого должно идти переменная или выражение, значение которой будет возвращено в вызываемый код. На коде ниже представлен метод, который запрашивает имя пользователя и сохраняет его в некоторую переменную:

using System;

using System.Collections.Generic;

using System.Data;

using System.IO;

using System.Linq;

using System.Reflection.Emit;

using System.Runtime.CompilerServices;

using System.Text;

using System.Threading;

using System.Threading.Tasks;

using System.Xml.Linq;

namespace ExampleApp

{

internal class Program

{

static void Main(string[] args)

{

string name = GetName();

Console.WriteLine($"Привет, {name}");

}

static string GetName()

{

Console.WriteLine("Введите свое имя:");

string name = Console.ReadLine();

return name;

}

}

}

Стоит обратить внимание на то, что переменная name объявляется в двух разных местах: в методе GetName и методе Main, однако конфликта имен не происходит. Это происходит потому, что фактически это две разные переменные и переменная из метода GetName будет доступна только внутри данного метода, такие переменные называются локальными. Переменная из метода Main также является локальной переменной, но ее область видимости отличается и поэтому конфликтов имен не происходит. Как и в случае с параметрами, тип данных, указанный в заголовке метода должен совпадать с тем типом данных, который будет у выражения или переменной, указанной после оператора return, иначе компилятор сообщит об ошибке приведения.

Иногда бывают ситуации, когда работа метода должна изменить несколько значений, т.е. фактически вернуть несколько значений из метода, но в операцию return можно записать только одно выражение или переменную.

В таких ситуациях можно использовать ключевые слова ref и out, однако использовать их без особой надобности не стоит, т.к. их использование может привести к ошибкам в работе приложения из-за особенностей их работы.

Первое ключевое слово ref используется для того, чтобы пометить им входной параметр метода. При пометке параметра этим ключевым словом, а делать это надо как при описании метода, так и при каждом его вызове приведет к тому, что изменение данного значения внутри метода будет отражаться на той переменной, которая была передана в качестве параметра.

Ниже приведен пример кода, который при передаче в него некоторого числа будет увеличивать его на единицу, а после этого будет возводить число в квадрат и возвращать результат:

using System;

using System.Collections.Generic;

using System.Data;

using System.IO;

using System.Linq;

using System.Reflection.Emit;

using System.Runtime.CompilerServices;

using System.Text;

using System.Threading;

using System.Threading.Tasks;

using System.Xml.Linq;

namespace ExampleApp

{

internal class Program

{

static void Main(string[] args)

{

int number = 5;

int res = IncAndPow(ref number);

Console.WriteLine(res); // 36

Console.WriteLine(number); // 6

}

static int IncAndPow(ref int value)

{

value++;

int result = value \* value;

return result;

}

}

}

Следующее ключевое слово используется, out, когда необходимо вернуть два и более значений из одного метода. Для иллюстрации его работы, вспомните, как работали методы преобразования TryParse. В них переменная, которая будет сохранять результат преобразования как раз таки помечается ключевым словом out. Стоит также сказать, что внутри метода должна инициализироваться эта переменная, иначе компилятор не сможет ее вернуть.

И последний ключевое слово, которое используется таким образом, in. Принцип действия полностью противоположен ключевому слову out. Дело в том, что если передавать в качестве параметра некоторый ссылочный тип данных, то при изменении его в коде метода эти изменения будут отражены везде. Чтобы избавиться от такой проблемы, используется ключевое слово in, которое запрещает изменять значение переменной где-либо еще, кроме как внутри данного метода.

Подробнее о ссылочных типах и типах-значениях будет рассказано в материале, посвященном объектно-ориентированному программированию.

И последнее, иногда бывают ситуации, когда заранее неизвестно, сколько значений необходимо передать внутрь метода. В таких ситуациях, используется ключевое слово params, которое указывает на то, что значений данной категории может быть от 0 и более, но точного значения не известно.

Параметр, объявленный с этим ключевым словом обязательно должен помечаться как одномерный массив и должен идти в списке объявления параметров последним. Все другие параметры должны идти перед этим массивом. После него никакие другие параметры указать нельзя. Однако, массивы не помеченные ключевым словом params мы также можем передавать в качестве аргументов. Отличие этих двух объявлений состоит в том, что массив, не помеченный ключевым словом params должен обязательно быть передан. Ниже представлен код, который иллюстрирует объявление такого метода:

int Sum(int initialValue: 0, params int[] numbers)

{

int result = initialValue;

foreach (var n in numbers)

{

result += n;

}

return result;

}

**5 Варианты заданий**

1. Дано натуральное число n. Найти среднее арифметическое n чисел, вводимых с клавиатуры.
2. Дано натуральное число n. Найти среднее геометрическое я чисел, вво­димых с клавиатуры.
3. Определить количество трехзначных натуральных чисел, сумма цифр которых равна заданному числу n.
4. Среди двухзначных чисел найти те, сумма квадратов цифр которых делиться на 13.
5. Написать программу поиска двухзначных чисел таких, что если к сумме цифр этого числа прибавить квадрат этой суммы, то получиться это число.
6. Написать программу поиска четырехзначного числа, которое при делении на 133 дает в остатке 125, а при делении на 134 дает в остатке 111.
7. Найти сумму целых положительных чисел из промежутка от a до b, кратных 4.
8. Найти сумму целых положительных чисел, больших 20, меньших 100, кратных 3 и заканчивающихся на 2, 4 или 8.
9. Дано натуральное число n. Получить все его натуральные делители.
10. Даны натуральные числа m и n. Получить все кратные им числа, меньшие m\*n.
11. Сумма цифр трехзначного числа кратна 7, само число также делиться на 7. Найти все такие числа.
12. Дано натуральное число n. Определить является ли оно палиндромом (перевертышем), с учетом четырех цифр. Например, палиндромами являются следующие числа: 1111, 6116, 0440.
13. Даны натуральные числа n и k. Из чисел от n до k выбрать те, запись которых содержит ровно три одинаковые цифры. Например, числа 6766, 5444 содержат ровно три одинаковые цифры.
14. Среди четырехзначных чисел выбрать те, у которых все четыре цифры различны.
15. Дано четырехзначное число n. Выбросить из его записи цифры 0 и 5, оставив прежним порядок остальных цифр. Например, из числа 1509 должно получиться число 19.
16. Натуральное число из n цифр является числом Армстронга, если сумма его цифр, возведенная в n – ю степень, равна самому числу (например, 153=13+53+33). Получить все трехзначные числа Армстронга.
17. Поменять местами первую и последнюю цифры числа.
18. Поменять порядок цифр числа на обратный. Например, было 12345, стало 54321.
19. Найти количество четных цифр целого положительного числа.
20. Найти самую большую цифру целого числа.
21. Найти сумму цифр целого числа, больших 5.
22. Сколько раз данная цифра k встречается в числе n.
23. Составить программу, проверяющую, является ли заданное натуральное число совершенным, то есть равным сумме своих положительных делителей, кроме самого этого числа.
24. Найти натуральное число от 1 до n с максимальной суммой делителей.
25. Даны натуральные числа a и b (a<b). Получить все простые числа из промежутка от a до b.
26. Даны натуральные числа n и m. Найти все пары дружественных чисел, лежащих в диапазоне от n до m. Два числа называются дружественными, если каждое из них равно сумме всех делителей другого (само число в качестве делителя не рассматривается).
27. Если мы сложим все цифры какого-либо числа, затем все цифры найденной суммы и будем повторять много раз, мы наконец получим однозначное число, называемое цифровым корнем данного числа. Например, цифровой корень числа 34697 равен 2 (3+4+6+9+7=29; 2+9=11; 1+1=2). Составить программу для нахождения цифрового корня натурального числа n.
28. Дано натуральное число. Составить программу разложения его на простые множители.
29. Найти все трехзначные числа, такие, что сумма его цифр равна a, а само число делиться на b.
30. Найти все трехзначные числа, такие, что сумма его цифр равна a, а само число оканчивается цифрой b.