**Лабораторная работа №4**

**Разработка методов, использование в программах встроенных функций**

**Цель работы:**

Познакомиться с методами, научиться использовать в программах встроенные функции с использованием среды **Visual Studio.Net.**

**1. Теоретические сведения**

Метод – это функциональный элемент класса, который реализует вычисления или другие действия, выполняемые классом или его экземпляром (объектом). Метод представляет собой законченный фрагмент кода, к которому можно обратиться по имени. Он описывается один раз, а вызываться может многократно. Совокупность методов класса определяет, что конкретно может делать класс. Например, стандартный класс Math содержит методы, которые позволяют вычислять значения математических функций.

Синтаксис объявления метода:

[атрибуты] [спецификторы] тип\_результата имя\_метода ([список\_формальных\_параметров])

{

тело\_метода;

return значение;

}

где:

1. *атрибуты* и *спецификторы* являются необязательными элементами в описанииметода. На данном этапе атрибуты нами использоваться не будут, а из всех

спецификаторов мы в обязательном порядке будем использовать спецификатор static, который позволяет обращаться к методу класса без создания экземпляра класса**.** Остальные спецификаторы мы рассмотрим в следующем модуле.

1. *тип\_результата* определяет тип значения,возвращаемого методом.Это можетбыть любой тип, включая типы классов, создаваемые программистом, а также тип void, который говорит о том, что метод ничего не возвращает.
2. *имя\_метода* будет использоваться для обращения к нему из других местпрограммы и должно быть корректно заданным с учетом требований, накладываемых на идентификаторы в С#.
3. *список\_формальных\_параметров* представляет собой последовательность пар,состоящих из типа данных и идентификатора, разделенных запятыми. Формальные параметры — это переменные, которые получают значения, передаваемые методу при вызове. Если метод не имеет параметров, то *список\_параметров* остается пустым.
4. *return* –это оператор безусловного перехода,который завершает работу метода ивозвращает *значение*, стоящие после оператора return, в точку его вызова. Тип *значения* должен соответствовать *типу\_\_результата,* или приводиться к нему.Еслиметод не должен возвращать никакого значения, то указывается тип void, и в этом случае оператор return либо отсутствует, либо указывается без возвращаемого значения.

Рассмотрим простейший пример метода:

class Program

{

static void Func() *//дополнительный метод*

{

Console.Write("x= ");

double x=double.Parse(Console.ReadLine());

double y = x\*x;

Console.WriteLine("y({0})={1}", x, y );

}

static void Main() *//точка входа в программу*

{

Func(); *//первый вызов метода* *Func* Func(); *//второй вызов метода* *Func*

}

}

В данном примере в метод Func не передаются никакие значения, поэтому список параметров пуст. Кроме того, метод ничего не возвращает, поэтому тип возвращаемого значения void, и в теле метода отсутствует оператор return. В основном методе Main мы вызвали метод Func два раза. При необходимости данный метод можно будет вызвать столько раз, сколько потребуется для решения задачи.

**Задания**.

1. Добавьте в метод Main третий вызов метода Func.
2. Преобразуйте программу так, чтобы метод Func вызывался n раз.

Изменим исходный пример так, чтобы в него передавалось значение х, а сам метод возвращал значение y.

class Program

{

static double Func( double x) *//дополнительный метод*

{

return x\*x; *//возвращаемое значение*

}

static void Main() *//точка входа в программу*

{

Console.Write("a=");

double a=double.Parse(Console.ReadLine()); Сonsole.Write("b=");

double b=double.Parse(Console.ReadLine()); Сonsole.Write("h=");

double h=double.Parse(Console.ReadLine()); for (double x = a; x <= b; x += h)

{

double y = Func(x); *//вызов метода* *Func* Console.WriteLine("y({0:f1})={1:f2}", x, y);

}

}

В данном примере метод Func содержит параметр х типа double. Для того, чтобы метод Func возвращал в вызывающий его метод Main значение выражения x\*x (типа double), перед именем метода указывается тип возвращаемого значения – double, а в теле метода используется оператор передачи управления – *return*. Оператор *return* завершает выполнение метода и передает управление в точку его вызова.

**Задания**.Преобразуйте программу так,чтобы метод*Func*возвращал значениевыражения:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |

1. x2+x-1 2. x2 при x>=0, 1/x при x<0

Рассмотрим другой пример:

2

class Program

{

static int Func( int x, int y) *// 1*

{

return (x>y)? x:y;

}

static void Main()

{

Console.Write("a=");

int a = int.Parse(Console.ReadLine()); Console.Write("b=");

int b = int.Parse(Console.ReadLine()); Console.Write("c=");

int c = int.Parse(Console.ReadLine());

int max = Func(Func(a, b), c); *//2 -* *вызовы метода* *Func* Console.WriteLine("max({0}, {1}, {2})={3}", a, b, c, max);

}

}

На этапе описания метода (строка 1) указывается, что метод Func имеет два целочисленных *формальных* параметра – x, y. На этапе вызова (строка 2) в метод передаются *фактические* параметры, которые по количеству и по типу совпадают с формальными параметрами. Если количество фактических и формальных параметров будет различным, то компилятор выдаст соответствующее сообщение об ошибке. Если параметры будут отличаться типами, то компилятор попытается выполнить неявное преобразование типов. Если неявное преобразование невозможно, то также будет сгенерирована ошибка.

Обратите внимание на то, что при вызове метода Func использовалось вложение одного вызова в другой.

**Задание**.Преобразуйте программу,не изменяя метод*Func,*чтобы можно было найтинаибольшее значение из четырех чисел: a, b, c, d.

Таким образом, параметры используются для обмена информацией между вызывающим и вызываемым методами. В С# предусмотрено четыре типа параметров: параметры-значения, параметры-ссылки, выходные параметры и параметры, позволяющие создавать методы с переменным количеством аргументов.

При передаче параметра *по значению* метод получает копии параметров, и операторы метода работают с этими копиями. Доступа к исходным значениям параметров у метода нет, а, следовательно, нет и возможности их изменить. Все примеры, рассмотренные ранее, использовали передачу данных по значению.

Рассмотрим небольшой пример:

class Program

{

static void Func(int x)

{

x += 10; *//* *изменили значение параметра*

Console.WriteLine("In Func: " + x);

}

static void Main()

{

int a=10;

Console.WriteLine("In Main: {0}", a); Func(a);

3

Console.WriteLine("In Main: {0}", a);

}

}



*Результат работы программы:*

In Main: 10

In Func: 20

In Main: 10



В данном примере значение формального параметра х было изменено в методе Func, но эти изменения не отразились на фактическом параметре *а* метода Main.

***Замечание.*** *Передача параметров по значению позволяет гарантировать целостность**исходных параметров только в случае, если исходные параметры имеют тип, являющийся размерным типом (value-type). Если исходный параметр имеет ссылочный тип (reference type), то в метод копируется ссылка на объект и, следовательно, можно менять значения объектов.*

При передаче параметров *по ссылке* метод получает копии адресов параметров, что позволяет осуществлять доступ к ячейкам памяти по этим адресам и изменять исходные значения параметров. Для того чтобы параметр передавался по ссылке, необходимо при описании метода перед формальным параметром и при вызове метода перед соответствующим фактическим параметром поставить спецификатор *ref*.

class Program

{

static void Func(int x, **ref** int y)

{

x += 10; y += 10; *//изменение параметров*

Console.WriteLine("In Func: {0}, {1}", x, y);

}

static void Main()

{

int a=10, b=10; *// 1*

Console.WriteLine("In Main: {0}, {1}", a,b); Func(a, **ref** b);

Console.WriteLine("In Main: {0}, {1}", a, b);

}

}



*Результат работы программы:*

In Main: 10 10

In Func: 20 20

In Main: 10 20



В данном примере в методе Func были изменены значения формальных параметров х и y. Эти изменения не отразились на фактическом параметре а, т.к. он передавался по значению, но значение b было изменено, т.к. он передавался по ссылке.

Передача параметра по ссылке требует, чтобы аргумент был инициализирован до вызова метода (см. строку 1).

**Задание**.Удалите в строке1инициализацию переменныхaиb,и попробуйтеоткомпилировать и запустить программу. Объясните, что произойдет.

Итак, при использовании неинициализированных фактических параметров компилятор выдаст сообщение об ошибке. Однако не всегда имеет смысл инициализировать параметр до вызова метода, например, если метод считывает

4

значение этого параметра с клавиатуры, или из файла. В этом случае параметр следует передавать как выходной, используя спецификатор *out.*.

class Program

{

static void Func(int x, **out** int y)

{

x += 10; y = x; *//* *определение значения выходного параметра* *y*

Console.WriteLine("In Func: {0}, {1}", x, y);

}

static void Main()

{

int a=10, b;

Console.WriteLine("In Main: {0}", a); Func(a, **out** b);

Console.WriteLine("In Main: {0}, {1}", a, b);

}

}



*Результат работы программы:*

In Main: 10

In Func: 20 20

In Main: 10 20



В данном примере в методе Func формальный параметр y и соответствующий ему фактический параметр b метода Main были помечены спецификатором out. Поэтому значение b до вызова метода Func можно было не определять, но изменение параметра повлекло за собой изменение значения параметра b.

***Замечание****.**Параметры,**позволяющие создавать методы с переменным количеством**аргументов, основаны на использовании массивов. Изучите данный прием работы с массивами самостоятельно.*

**Варианты задания 1:**

|  |  |  |  |
| --- | --- | --- | --- |
|  | Решение простейших задач с использованием вспомогательных методов: | |  |
| 1. | Разработать метод min(a,b) для нахождения минимального из | двух | чисел. |
|  | Вычислить с помощью него значение выражения z=min(3x,2y)+min(x-y,x+y). | |  |
| 2. | Разработать метод max(a,b) для нахождения максимального из | двух | чисел. |
|  | Вычислить с помощью него значение выражения z=max(x,2y-x)+max(5x+3y,y). | | |

1. Разработать метод f(x), который вычисляет значение по следующей формуле: f(x)=x3-sin x. Определить, в какой из точек а или b, функция принимает наибольшее значение.
2. Разработать метод f(x), который возвращает вторую справа цифру натурального числа x. Вычислить с помощью него значение выражения z=f(a)+f(b)-f(c).
3. Разработать метод f(x), который нечетное число заменяет на 0, а четное число уменьшает в два раза. Продемонстрировать работу данного метода на примере.
4. Разработать метод f(x), который в двузначном числе меняет цифры местами, а остальные числа оставляет без изменения. Продемонстрировать работу данного метода на примере.
5. Разработать метод f(x), который в трехзначном числе меняет местами первую с последней цифрой, а остальные числа оставляет без изменения. Продемонстрировать работу данного метода на примере.
6. Разработать метод f(x1, y1, x2, y2), который вычисляет длину отрезка по координатам вершин (x1, y1) и (x2, y2), и метод d(a, b, c), который вычисляет периметр треугольника по длинам сторон a, b, c. С помощью данных методов найти периметр треугольника, заданного координатами своих вершин.
7. Разработать метод f(x1, y1, x2, y2), который вычисляет длину отрезка по координатам вершин (x1, y1) и (x2, y2), и метод max(a, b), который вычисляет максимальное из чисел a, b. С помощью данных методов определить, какая из трех точек на плоскости наиболее удалена от начала координат.
8. Разработать метод f(x1, y1, x2, y2), который вычисляет длину отрезка по координатам вершин (x1, y1) и (x2, y2), и метод min(a, b), который вычисляет минимальное из чисел a, b. С помощью данных методов найти две из трех заданных точек на плоскости, расстояние между которыми минимально.
9. Разработать метод f(x), который четное число заменяет на 1, а к нечетному числу добавляет 2. Продемонстрировать работу данного метода на примере.

**Варианты задания 2:**

1. Вводится последовательность целых чисел, 0 – конец последовательности. Для каждого числа последовательности найти количество его делителей (функцией оформить определение количества делителей числа).

2. Вводится последовательность целых чисел, 0 – конец последовательности. Для каждого числа последовательности найти сумму его простых делителей (функцией оформить определение суммы простых делителей числа).

3. Вводится последовательность из N целых чисел. Найти наименьшую по значению цифру в каждом числе (функцией оформить определение наименьшей цифры числа).

4. Вводится последовательность из N целых чисел. Найти наибольшую по значению четную цифру в каждом числе (функцией оформить определение наибольшей четной цифры числа).

5. Вводится последовательность из N целых чисел. Найти в каждом числе сумму четных цифр (функцией оформить определение суммы четных цифр числа).

6. Вводится последовательность из N целых чисел. Найти в каждом числе количество четных и нечетных цифр (функциями оформить определение количества четных и нечетных цифр числа).

7. Вводится последовательность из N целых чисел. Найти среднее арифметическое его цифр (функцией оформить определение среднего арифметического цифр числа).

8. Вводится последовательность целых чисел, 0 – конец последовательности. Для каждого числа последовательности проверить, представляют ли его цифры строго возрастающую последовательность, например, 3456 (результатом функции будет 1 – Да, 0 - НЕТ).

9. Вводится последовательность целых чисел, 0 – конец последовательности. Для каждого числа последовательности проверить, представляют ли его цифры строго убывающую последовательность, например, 1234 (результатом функции будет 1 – Да, 0 - НЕТ).

10.Вводится последовательность из N целых чисел. Найти количество двух- и количество трехразрядных чисел в последовательности (функцией оформить определение количества разрядов числа).

11.Вводится последовательность из N целых чисел. Для каждого числа последовательности вывести новое число, которое получится после записи цифр числа в обратном порядке (функцией оформить определение нового числа для заданного).

12.Найти р!, где р – каждое третье простое число в диапазоне от n1 до n2 (функциями оформить определение факториала и проверку, является ли число простым).

13.Дано натуральное число N. Определить M=N!. Проверить, как изменилось количество разрядов в числе M по сравнению с количеством разрядов числа N (функцией оформить определение количества разрядов числа)

14.Дано натуральное число N. Уменьшить число в 2 раза (деление нацело). Проверить, изменилось ли после уменьшения количество разрядов в числе (функцией оформить определение количества разрядов числа).

15.Дано натуральное число N, проверить, простое оно или нет. Увеличить его значение на натуральное число M. Проверить, осталось ли оно простым (функция возвращает 1, если число простое, 0 – в противном случае).

16.Вводится последовательность из N целых чисел. Сформировать новую последовательность, каждый элемент которой равен 1, если соответствующее число простое, 2 – если число совершенное, 0 – в остальных случаях. Вычислить количество 0, 1 и 2 в выходной последовательности (функциями оформить проверку является ли число простым, совершенным).

17.Вводится последовательность из N целых чисел. Для каждого числа последовательности определить минимальную цифру и ее месторасположение в числе (функция определяет количество цифр в числе).

18.Вводится последовательность из N целых чисел. Найти максимальное совершенное число в последовательности, если таких несколько, вывести их количество (результатом функции будет 1 – число совершенное, 0 - нет).

19.Вводится последовательность целых чисел, 0 – конец последовательности. Определить среднее арифметическое простых чисел последовательности (результатом функции будет 1 – число простое, 0 - нет).

20.Вводится последовательность целых чисел, 0 – конец последовательности. Найти количество совершенных и простых чисел в последовательности (результатами функций будет:1 – число простое, 0 – число непростое; 1 – число совершенное, 0 - нет).

21.Вводится последовательность целых чисел, 0 – конец последовательности. Определить минимальное простое число последовательности (результатом функции будет 1 – число простое, 0 - нет).

22.Вводится последовательность из N целых чисел. Каждое простое число последовательности увеличить в два раза, посчитать количество простых чисел в исходной последовательности (результатом функции будет 1 – число простое, 0 - нет).

23. Вводится последовательность целых чисел, 0 – конец последовательности. Для каждого числа последовательности найти количество его делителей (функцией оформить определение количества делителей числа).