Лекция 10. Пользовательские функции

Цель лекции: Формирование у студентов знаний, связанных с разработкой пользовательских функций.

В результате изучения данной лекции студенты будут способны:

- Продемонстрировать знание синтаксиса создания пользовательских функций
- Продемонстрировать понимание необходимости использования пользовательских функций
- Продемонстрировать понимание синтаксиса использования параметров в методах

- *Метод* представляет собой законченный фрагмент кода, к которому можно обратиться по имени. Он описывается один раз, а вызываться может столько раз, сколько необходимо. Один и тот же метод может обрабатывать различные данные, переданные ему в качестве аргументов.

- Упрощение программ с многократным использованием кода привело к использованию подпрограмм. *Подпрограмма* – поименованная часть компьютерной программы, содержащая описание определенного набора действий. Подпрограмма может быть многократно вызвана из разных частей программы. В различных языках программирования подпрограммы могут разделяться на процедуры и функции. *Процедура* выполняет действие, но не возвращает значение, функция же возвращает значение. В C# все подпрограммы являются функциями, но использование типа void (англ. – пусто) позволяет не возвращать никаких значений. Т.к. функции принадлежат классам, эти функции называются *методами*.

Синтаксис определения метода:

доступ возвращаемый_тип имя_метода (список_параметров)
{
// Тело метода
}
где

доступ – необязательный модификатор доступа. Отсутствие подразумевает private, определяет закрытый метод, доступный только внутри класса. Открытый модификатор public позволяет вызывать метод из любой другой части программы. Программы текущей работы выполняются внутри одного класса, поэтому этот модификатор будет пропущен.

возвращаемый_тип определяет, значение какого типа вычисляется с помощью метода. Часто употребляется термин «метод возвращает значение», поскольку после выполнения метода происходит возврат в то место вызывающей функции, откуда был вызван метод, и передача туда значения выражения, записанного в операторе return. Если метод не возвращает никакого значения, в его заголовке задается тип void, а оператор return отсутствует.

список_параметров – последовательность пар, состоящих из типа и идентификатора и разделенных запятыми. При отсутствии параметров список остается пустым. Параметры, задаваемые в описании метода, также называются *формальными параметрами* (или просто параметрами).
Параметры, используемые при вызове метода, называются *фактическими параметрами* или *аргументами*.

При вызове подпрограммы сохраняется адрес оператора, следующего за ее вызовом, и управление передается операторам подпрограммы. Значения формальных параметров, как поля бланка, заполняются значениями аргументов. После завершения выполнения подпрограммы управление переходит к оператору, адрес которого был сохранен перед вызовом. Если тип метода отличен от void и метод вызывается внутри выражения, то в место вызова метода подставляется возвращаемое значение. Если метод вызывается не в выражении, то возвращаемое значение игнорируется.

Константы и переменные, заданные вне методов в классе доступны во всех методах класса (и из экземпляров других классов, если описаны с модификатором public) и являются для методов класса *глобальными*.
Переменные (и параметры) внутри методов являются *локальными* и теряются при выходе из метода. Более того, переменные могут быть локальными внутри программного блока {} в теле метода.
Примечание. Метод, не возвращающий значение, вызывается отдельным оператором, а метод, возвращающий значение, – в составе выражения в правой части оператора присваивания.
Параметры методов
Рассмотрим более подробно, каким образом метод обменивается информацией с вызвавшим его кодом. При вызове метода выполняются следующие действия:
1. Вычисляются выражения, стоящие на месте аргументов.
2. Выделяется память под параметры метода в соответствии с их типом.
3. Каждому из параметров сопоставляется соответствующий аргумент (аргументы как бы накладываются на параметры и замещают их).
4. Выполняется тело метода.
5. Если метод возвращает значение, оно передается в точку вызова;
если метод имеет тип void, управление передается на оператор, следующий после вызова.

При этом проверяется соответствие типов аргументов и параметров и при необходимости выполняется их преобразование. При несоответствии типов выдается диагностическое сообщение.

Примечание. Главное требование при передаче параметров состоит в том, что аргументы при вызове метода должны записываться в том же порядке, что и параметры в заголовке метода. Количество аргументов должно соответствовать количеству параметров. На имена никаких ограничений не накладывается: имена аргументов могут как совпадать, так и не совпадать с именами параметров.

Задача 10.1. Написать метод, который вычисляет площадь поверхности параллелепипеда

*S = (L*w+ L*h + w*h)*2*

Программный код:

```
using System;
namespace C52a
{
class Program
{
// площадь поверхности параллелепипеда
static double ps(double L, double w, double h)
{
return (L * w + L * h + w * h) * 2;
}
static void Main(string[] args)
{
```

```
double L, w, h; // длина, ширина и высота параллелепипеда
double s; // площадь поверхности параллелепипеда
Console.WriteLine(" Вычисление площади поверхности
параллелепипеда\n ");
Console.WriteLine(" Введите исходные данные:");
Console.Write(" Длина (см) -> ");
L = Convert.ToDouble(Console.ReadLine());
Console.Write(" Ширина (см) -> ");
```

```
w = Convert.ToDouble(Console.ReadLine());
Console.Write(" Высота (см) -> ");
h = Convert.ToDouble(Console.ReadLine());
s = ps(L, w, h);
Console.WriteLine(" \nПлощадь поверхности = {0,5:F2}
кв. см.", s);
Console.ReadLine();
}
```

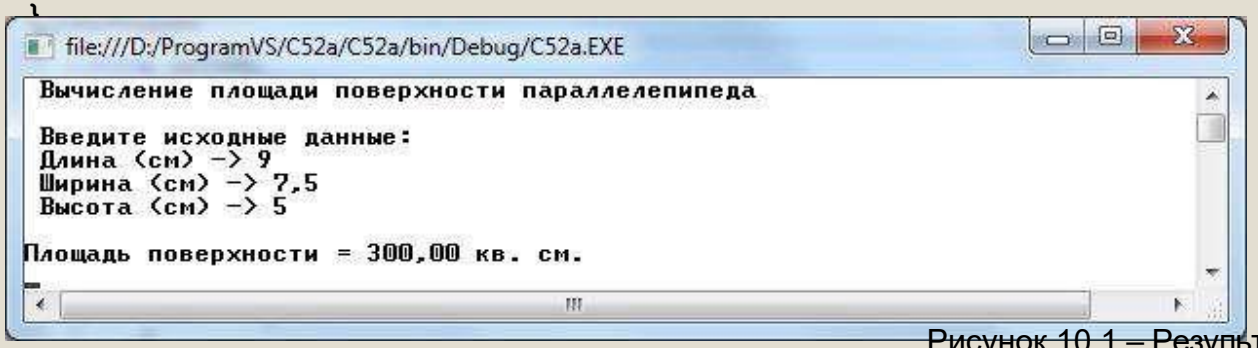

Рисунок 10.1 – Результаты решения задачи 10.1

Способы передачи параметров

Существуют два способа передачи параметров: по значению и по ссылке. *При передаче по значению* метод получает копии значений аргументов, и операторы метода работают с этими копиями. Доступа к исходным значениям аргументов у метода нет, а следовательно, нет и возможности их изменить.

При передаче по ссылке (*по адресу*) метод получает копии адресов аргументов, он осуществляет доступ к ячейкам памяти по этим адресам и может изменять исходные значения аргументов, модифицируя параметры.

В C# для обмена данными между вызывающей и вызываемой функциями предусмотрено четыре типа параметров:

- параметры - значения;
- параметры - ссылки – описываются с помощью ключевого слова ref;
- выходные параметры – описываются с помощью ключевого слова out;
- массив параметров – описывается с помощью ключевого слова params.

Ключевое слово предшествует описанию типа параметра. Если оно опущено, параметр считается параметром- значением. Массив параметров может быть только один и должен располагаться последним в списке, например:

public int Calculate(int a, ref int b, out int c, params int[] d)…

Параметры – значения

Параметр - значение описывается в заголовке метода следующим образом:
тип имя
Пример заголовка метода, имеющего один параметр- значение целого типа:
void P(int x)
Параметр x представляет собой локальную переменную, которая получает свое значение из вызывающей функции при вызове метода. В метод передается копия значения аргумента.
Механизм передачи следующий: из ячейки памяти, в которой хранится переменная, передаваемая в метод, берется ее значение и копируется в специальную область памяти – область параметров. Метод работает с этой копией, следовательно, доступа к ячейке, где хранится сама переменная, не имеет. По завершении работы метода область параметров освобождается. Таким образом, для параметров- значений используется *передача по значению*. Этот способ годится только для величин, которые не должны измениться после выполнения метода, то есть для его исходных данных.

Параметры – ссылки

Во многих методах все величины, которые метод должен получить в качестве исходных данных, описываются в списке параметров, а величина, которую вычисляет метод как результат своей работы, возвращается в вызывающий код с помощью оператора return. Очевидно, что если метод должен возвращать более одной величины, такой способ не годиться. Еще одна проблема возникает, если в методе требуется изменить значение каких- либо передаваемых в него величин. В этих случаях используются *параметры - ссылки.*

Признаком параметра - ссылки является ключевое слово ref перед описанием параметра:

ref тип имя

Пример заголовка метода, имеющего один параметр - ссылку целого типа:

void P(ref int x)

При вызове метода в область параметров копируется не значение аргумента, а его адрес, и метод через него имеет доступ к ячейке, в которой хранится аргумент. Таким образом, параметры - ссылки передаются *по адресу* (чаще употребляется термин «передача по ссылке»). Метод работает непосредственно с переменной из вызывающей функции и, следовательно, может ее изменить, поэтому если в методе требуется изменить значения параметров, они должны передаваться только по ссылке.

Задача 10.2. Использование параметра-ссылки

Программный код:

```
using System;
namespace C47
{
class Program
{
static void P(int a, ref int b)
{
a = 44; b = 33;
Console.WriteLine(" внутри метода {0} {1}", a, b);
}
static void Main(string[] args)
{
int a = 2, b = 4;
Console.WriteLine(" до вызова {0} {1}", a, b);
P(a, ref b);
Console.WriteLine(" после вызова {0} {1}", a, b);
Console.ReadLine();
}
}
}
```



Результаты расчета см. рис. 10.2.

Рисунок 10.2 – Результаты решения задачи 10.2

В этом примере значение переменной a в функции Main не изменилось, поскольку переменная передавалась по значению, а значение переменной b изменилось потому, что она была передана по ссылке.

Заключение

В рамках данной лекции были изучены основы разработки и применения пользовательских функций (методов) в языке C#: принципы декомпозиции программ на подпрограммы, синтаксис объявления и вызова методов, различия между методами, возвращающими значение, и методами типа `void`, а также правила области видимости переменных и механизмы передачи параметров (по значению и по ссылке с использованием модификаторов `ref`, `out`, `params`).

Использование пользовательских методов является ключевым приемом структурного и объектно-ориентированного программирования. Оно позволяет избежать дублирования кода, разбить сложную алгоритмическую задачу на логически завершенные и легко тестируемые модули, а также сделать программу понятной для сопровождения. Понимание разницы между передачей аргументов по значению и по ссылке обеспечивает полный контроль над тем, какие данные остаются защищенными от изменений внутри метода, а какие могут модифицироваться или возвращаться в качестве нескольких результатов вычислений.

Вопросы для самопроверки

1. **Понятие метода:** Что такое метод (подпрограмма) в языке C# и какие преимущества дает разбиение программы на пользовательские функции? Чем метод с возвращаемым типом отличается от метода, объявленного с ключевым словом `void`?

2. **Синтаксис и вызов:** Из каких обязательных и необязательных элементов состоит заголовок метода? Как в коде программы вызывается метод типа `void` и как — метод, возвращающий конкретное значение?

3. **Параметры и аргументы:** В чем заключается разница между формальными параметрами и фактическими параметрами (аргументами)? Какие требования предъявляются к порядку, количеству и типам аргументов при вызове метода?

4. **Область видимости:** Чем локальные переменные, объявленные внутри метода, отличаются от глобальных полей, объявленных на уровне класса? Что происходит с локальными переменными после завершения работы метода?

5. **Передача по значению:** Как работает механизм передачи параметров по значению? Изменится ли значение исходной переменной в вызывающем коде, если внутри метода изменить соответствующий ей параметр-значение?

6. **Передача по ссылке и модификаторы:** Для решения каких задач применяются параметры-ссылки (`ref`) и выходные параметры (`out`), если метод должен вернуть несколько результатов?

Список используемых источников

1. **Шилдт Г.** C# 4.0: полное руководство. — М.: Вильямс, 2019. — 1056 с. *(Глава 6: Введение в классы, объекты и методы; Глава 8: Подробное рассмотрение методов и классов — передача аргументов по значению и по ссылке, `ref`, `out`, `params`).*

2. **Троелсен Э., Джепикс Ф.** Язык программирования C# и платформа .NET. — СПб.: Диалектика, 2022. — 1312 с. *(Раздел: Методы и модификаторы параметров в C#).*

3. **Албахари Д., Албахари Б.** C#. Справочник. Полное описание языка. — СПб.: БХВ-Петербург, 2023. — 1088 с. *(Спецификация методов, локальных переменных и способов передачи параметров).*

4. **Павловская Т. А.** C#. Программирование на языке высокого уровня: Учебник для вузов. — СПб.: Питер, 2020. — 432 с. *(Глава: Подпрограммы, методы и механизмы обмена данными).*

5. **Официальная документация Microsoft Learn:** Методы (руководство по программированию на C#) [Электронный ресурс]. — URL: microsoft.com (дата обращения: 13.11.2026).