*Методы*

представляют собой подпрограммы, которые манипулируют данными, определенными

в классе, а во многих случаях они предоставляют доступ к этим данным. Как правило,

другие части программы взаимодействуют с классом посредством его методов.

Метод состоит из одного или нескольких операторов. В грамотно написанном коде

C# каждый метод выполняет только одну функцию. У каждого метода имеется свое

имя, по которому он вызывается. В общем, методу в качестве имени можно присвоить

любой действительный идентификатор. Следует, однако, иметь в виду, что идентификатор

Main() зарезервирован для метода, с которого начинается выполнение программы.

Кроме того, в качестве имен методов нельзя использовать ключевые слова С#.

Ниже приведена общая форма определения метода:

**доступ возращаемый\_тип имя(список\_параметров) {**

**// тело метода**

**}**

где *доступ* — это модификатор доступа, определяющий те части программы, из которых

может вызываться метод. Как пояснялось выше, указывать модификатор доступа

необязательно. Но если он отсутствует, то метод оказывается закрытым (***private***)

в пределах того класса, в котором он объявляется. Полный код в glava6\_2.

class Building

{

public int Floors;

public int Area;

public int Occupants;

//area for one person

public void AreaPerPerson()

{

Console.Write(Area / Occupants + " for one person");

}

}

В этой строке объявляется метод, именуемый AreaPerPerson и не имеющий параметров.

Для него указывается тип public, а это означает, что его можно вызывать из любой

другой части программы. Метод AreaPerPerson() возвращает пустое значение типа

void, т.е. он практически ничего не возвращает вызывающей части программы. Анализируемая

строка завершается фигурной скобкой, открывающей тело данного метода.

**Возврат из метода**

В отличии от С++ в С# из методов с типом void можно использовать return для немедленного завершения метода.Достаточно воспользоваться следующей формой оператора return.

***return;***

Когда выполняется этот оператор, управление возвращается вызывающей части программы, а оставшийся в методе код пропускается.

public void MyMeth()

{

int i;

for (i = 0; i < 10; i++)

{

if (i == 5) return; //break step 5

Console.WriteLine();

}

}

В данном примере выполняется лишь 5 полноценных шагов цикла for, поскольку при значении 5 переменной ***i*** происходит возврат из метода.

В методе допускается наличие нескольких операторов return, особенно если имеются

два или более вариантов возврата из него.

Итак, напомним еще раз: возврат из метода типа void может произойти при двух

условиях: по достижении закрывающей фигурной скобки или при выполнении оператора

***return***.

**Возврат значения**

Для возврата значения из метода в вызывающую часть программы служит следующая

форма оператора return:

***return значение;***

где ***значение***— это конкретное возвращаемое значение.

class Building

{

public int Floors;

public int Area;

public int Occupants;

//area for one person

public int AreaPerPerson()

{

return Area / Occupants;

}

}

Весь код в листинге glava6\_3.

**Использование параметров**

При вызове метода ему можно передать одно или несколько значений. Значение,

передаваемое методу, называется *аргументом.* А переменная, получающая аргумент,

называется *формальным параметром,* или просто *параметром.* Параметры объявляются

в скобках после имени метода. (glava6\_4)

class ChkNum

{

//return true if x = simple num

public bool IsPrime(int x)

{

if (x <= 1) return false;

for (int i = 2; i <= x / i; i++)

if ((x % i) == 0) return false;

return true;

}

}

У метода может быть не только один, но и несколько параметров. Каждый его параметр

объявляется, отделяясь от другого запятой. (glava6\_4.1)

class ChkNum

{

//return true if x = simple num

public bool IsPrime(int x)

{

if (x <= 1) return false;

for (int i = 2; i <= x / i; i++)

if ((x % i) == 0) return false;

return true;

}

//return min multiply

public int LeastComFactor(int a, int b)

{

int max;

if (IsPrime(a) || IsPrime(b)) return 1;

max = a < b ? a : b;

for (int i = 2; i <= max / 2; i++)

if (((a % i) == 0) && ((b % i) == 0)) return i;

return 1;

}

}

**Добавление параметризированного метода**

С помощью параметризированного метода можно дополнить класс ***Building*** новым

средством, позволяющим вычислять максимальное количество жильцов в здании,

исходя из определенной величины минимальной площади на одного человека. (glava6\_5)

class Building

{

public int Floors;

public int Area;

public int Occupants;

//area per person

public int AreaPerPerson()

{

return Area / Occupants;

}

//Maximum of ppl in building

//according to min area for one person

public int MaxOccupant(int minArea)

{

return Area / minArea;

}

}