Метод может возвратить данные любого типа, в том числе и тип класса. Ниже в качестве

примера приведен вариант класса Rect, содержащий метод Enlarge(), в котором

строится прямоугольник с теми же сторонами, что и у вызывающего объекта

прямоугольника, но пропорционально увеличенными на указанный коэффициент.

class Rect

{

int width;

int height;

…

//method returns square with sides

//larger by typed values

public Rect Enlarge(int factor)

{

return new Rect(width\*factor, height \* factor);

}

}

static void Main()

{

Rect r1 = new Rect(4,5);

Console.Write("Size of square r1: ");

r1.Show();

Console.WriteLine("Area of square r1: " + r1.Area());

Console.WriteLine();

//make sqr two times larger than r1

Rect r2 = r1.Enlarge(2);

Console.Write("Size of square r2: ");

r2.Show();

Console.WriteLine("Area of square r2: " + r1.Area());

}

Когда метод возвращает объект, последний продолжает существовать до тех пор,

пока не останется ссылок на него. После этого он подлежит сборке как "мусор". Следовательно,

объект не уничтожается только потому, что завершается создавший его

метод.

Одним из практических примеров применения возвращаемых данных типа объектов

служит ***фабрика класса****,* которая представляет собой метод, предназначенный

для построения объектов его же класса. В ряде случаев предоставлять пользователям

класса доступ к его конструктору нежелательно из соображений безопасности или же

потому, что построение объекта зависит от некоторых внешних факторов. В подобных

случаях для построения объектов используется фабрика класса. (***glava8\_8.1***)

class MyClass

{

int a, b; //private

//make class factory

public MyClass Factory(int i, int j)

{

MyClass t = new MyClass();

t.a = i;

t.b = j;

return t;

}

}

static void Main()

{

MyClass ob = new MyClass();

int i, j;

//make objs using factory of class

for (i = 0, j = 10; i < 10; i++, j--)

{

MyClass anotherOb = ob.Factory(i, j); //create obj

anotherOb.Show();

}

Console.WriteLine();

}

На каждом шаге итерации цикла создается переменная ссылки на объект

anotherOb, которой присваивается ссылка на объект, формируемый фабрикой класса.

По завершении каждого шага итерации цикла переменная anotherOb выходит за

пределы области своего действия, а объект, на который она ссылается, утилизируется.

**Возврат массива из метода**

В C# массивы реализованы в виде объектов, а это означает, что метод может также

возвратить массив. (В этом отношении C# отличается от C++, где не допускается возврат

массивов из методов.) В качестве примера ниже приведена программа, в которой

метод FindFactors() возвращает массив, содержащий множители переданного ему

аргумента. (***glava8\_9***)

class Factor

{

//Method will return array facts, it contains factor of num argument

//numfactor using out will return quantity of factors

public int[] FindFactors(int num, out int numfactors)

{

int[] facts = new int[80];

int i, j;

//find factors and get into array facts

for (i = 2, j = 0; i < num / 2 + 1; i++)

if ((num % i) == 0)

{

facts[j] = i;

j++;

}

numfactors = j;

return facts;

}

}

static void Main()

{

Factor f = new Factor();

int numfactors;

int[] factors;

factors = f.FindFactors(1000, out numfactors);

Console.WriteLine("Factors of num 100: ");

for (int i = 0; i < numfactors; i++)

Console.Write(factors[i] + " ");

Console.WriteLine();

}

Обратите внимание на то, как указывается возвращаемый массив типа int. Этот

синтаксис можно обобщить. Всякий раз, когда метод возвращает массив, он указывается

аналогичным образом, но с учетом его типа и размерности. Например, в следующей

строке кода объявляется метод someMeth(), возвращающий двумерный массив

типа double.

***public double[,] someMeth() { // ...***