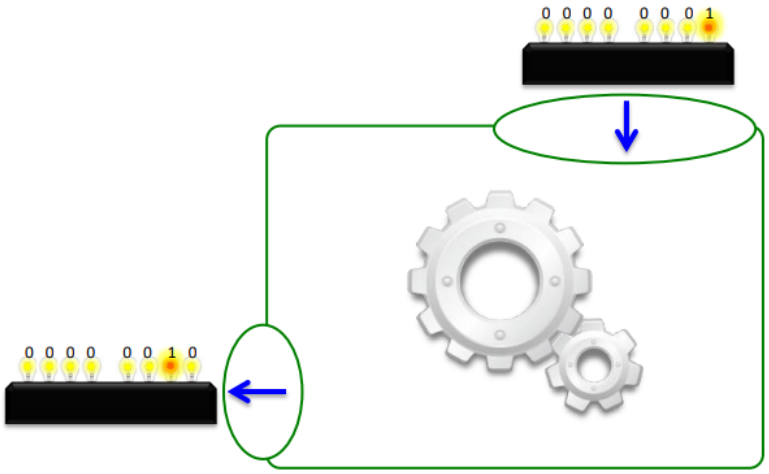
Лабораторная работа № 6

МЕТОДЫ

Метод

Method

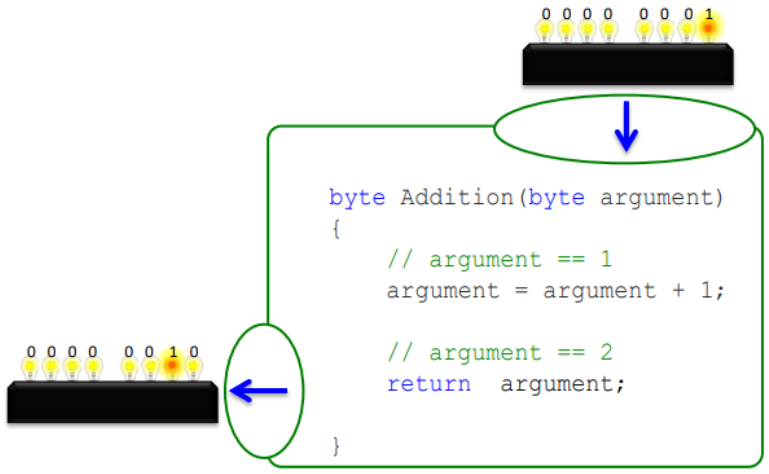
Метод — это именованная часть программы, которая может вызываться из других частей программы столько раз, сколько необходимо.



Метод

Method

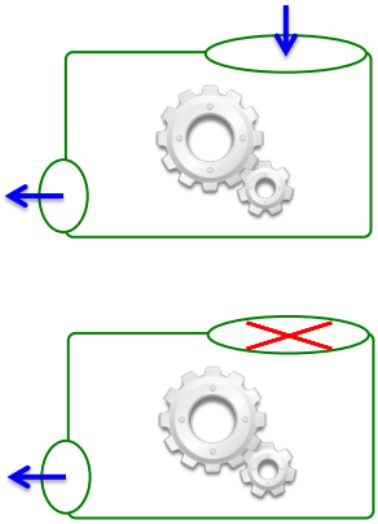
Метод — это именованная часть программы, которая может вызываться из других частей программы столько раз, сколько необходимо.



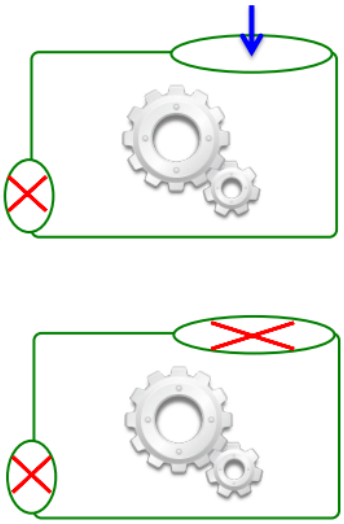
Методы

Функции и Процедуры

Функции – возвращают значения



Процедуры – не возвращают значений



Метод

Сигнатура и Тело

Методы

Создание и вызов

Для создания метода необходимо:

1) Указать тип возвращаемого значения, если метод ничего не возвращает указать `void`.
2) Выбрать имя метода в соответствии с действием, которое он выполняет.
3) Если метод принимает аргументы – обязательно указать их тип и имя, если нет – оставить аргументные скобки () пустыми.
4) Если метод имеет возвращаемое значение, обязательно в теле метода должно присутствовать ключевое слово `return`. Тип возвращаемого значения метода должен соответствовать типу значения, указанному после ключевого слова return.

Для вызова метода необходимо:

1) Написать имя метода.
2) Обязательно указать после имени аргументные скобки(), если метод принимает какие-то аргументы, передать соответствующее количество аргументов соответствующего типа.

Примеры : Методы

Проект 1. Создайте проект с работой метода – процедурой

```
using System;

// Методы (Процедуры).

namespace Methods
{
   class Program
   {
      // На 12-й строке, создаем метод с именем Procedure, который ничего не принимает и ничего не возвращает.
      // В теле метода, на 14-й строке выводим на экран строку - Hello!

      static void Procedure()
      {
         Console.WriteLine("Hello!");
      }

      static void Main()
      {
         // В теле метода Main на 21-й строке, вызываем метод Procedure.

         Procedure();

         // Delay.
         Console.ReadKey();
      }
   }
}
```

Проект 2.

Метод – функция

```
using System;

// Методы (Функции).

namespace Methods
{
   class Program
   {
      // На 12-й строке, создаем метод с именем Function, который ничего не принимает и возвращает строковое значение.
      // В теле метода, используя ключевое слово return, возвращаем строку - Hello!

      static string Function()
      {
         return "Hello!";
      }

      static void Main()
      {
         // В теле метода Main на 22-й строке, создаем строковую локальную переменную с именем @string
         // и присваиваем ей возвращаемое значение метода Function.

         string @string = Function();

         Console.WriteLine(@string);

         // Delay.
         Console.ReadKey();
      }
   }
}
```

Проект 3.

```
using System;

// Методы (Функции).

namespace Methods
{
   class Program
   {
      static string Function()
      {
         string word = "Hello!";

         return word;
      }

      static void Main()
      {
         string word = Function();

         Console.WriteLine(word);

         // Delay.
         Console.ReadKey();
      }
   }
}
```

Проект 4.

Использование параметров

```
using System;

namespace _005_Methods
{
// Методы (Функции).

        class Program
    {
        /// <summary>
        /// Сложение двух целых чисел.
        /// </summary>
        /// <param name="summand1">Первое слагаемое</param>
        /// <param name="summand2">Второе слагаемое</param>
        /// <returns>Сумма</returns>
        static int Add(int summand1, int summand2)
        {
            return summand1 + summand2;
        }

        static void Main()
        {
            int summand1 = 2, summand2 = 3;

            int sum = Add(summand1, summand2);

            Console.WriteLine("{0} + {1} = {2}", summand1, summand2, sum);

            // Delay.
            Console.ReadKey();
        }
    }

}
```

Проект 5. Работа с логическими типами данных

```
using System;

// Методы (Функции).

namespace Methods
{
    class Program
    {
        // Методы, которые возвращают логическое значение, называют методами-предикатами.

        static bool And(bool a, bool b)
        {
            return a && b;
        }

        static void Main()
        {
            bool operand1 = true, operand2 = true;

            bool result = And(operand1, operand2);

            Console.WriteLine("{0} && {1} = {2}", operand1, operand2, result);

            // Delay.
            Console.ReadKey();
        }
    }
}
```

Методы

Методы – набор операторов, которые выполняют определенные действия. Общее определение методов выглядит следующим образом:

```
[модиф-ры] тип_возвр._зн. назв._мет. ([параметры])
{// тело метода}
```

Рассмотрим на примере метода Main:

```
static void Main(string[] args)
 {Console.WriteLine("привет мир!");}
```

Ключевое слово `static` является модификатором. Далее идет тип возвращаемого значения. В данном случае ключевое слово `void` указывает на то, что метод ничего не возвращает. Такой метод еще называется процедурой. Далее идет название метода – `Main`, и в скобках параметры – `string[] args`. В фигурные скобки заключено тело метода – все действия, которые он выполняет.

Чтобы вызвать метод в программе, надо указать имя метода, а после него в скобках значения для его параметров:

```
string message = Hello(); // вызов первого метода.
```

Функции

В отличие от процедур функции возвращают определенное значение. Например:

```
int Factorial()
 {return 1;}
```

В функции в качестве типа возвращаемого значения вместо `void` используется любой другой тип. В данном случае тип `int`. Функции также отличаются тем, что мы обязательно должны использовать оператор `return`, после которого ставится возвращаемое значение. Также стоит отметить, что возвращаемое значение всегда должно иметь тот же тип, что значится в определении функции.

Параметры методов

Параметры представляют собой переменные, которые определяются в сигнатуре метода и создаются при его вызове. В C# для обмена данными между вызывающей и вызываемой функциями предусмот-

рено четыре типа параметров:

– *параметры-значения*;

– *параметры-ссылки* – описываются с помощью ключевого слова `ref`;

– *выходные параметры* – описываются с помощью ключевого слова `out`:

– *параметры-массивы* – описываются с помощью ключевого слова `params`.

Ключевое слово предшествует описанию типа параметра. Если оно опущено, параметр считается параметром-значением. Параметр-массив 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 ) ...
```

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. Параметр-значение описывается в заголовке метода следующим образом:

```
тип имя
```

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

```
void P(int x)
```

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

```
ref тип имя
```

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

```
void P(ref int x)
```

Выходные параметры. Выше мы использовали входные параметры. Но параметры могут быть также выходными. Чтобы сделать параметр выходным, перед ним ставится модификатор `out`:

```
static void Sum(int x, int y, out int a)
 {a = x + y;}
```

Здесь результат возвращается не через оператор `return`, а через выходной параметр. Использование в программе:

```
int x = 10; int z;
   Sum(x, 15, out z);
```

Причем, как и в случае с `ref`, ключевое слово `out` используется как при определении метода, так и при его вызове.

Также обратите внимание, что методы, использующие такие параметры, обязательно должны присваивать им определенное значение.

Кортежи

Массивы комбинируют объекты одного типа, а кортежи (`tuple`) могут комбинировать объекты различных типов. Кортежи предоставляют удобный способ для работы с набором значений, который был добавлен в версии C# 7.0. *Кортеж* представляет набор значений, заключенных в круглые скобки:

```
var tuple = (5, 10);
```

В данном случае определен кортеж `tuple`, который имеет два значения: 5 и 10. Обращаться к каждому из элементов кортежа можно через поля с названиями `Item[поряд._номер_поля_в_кортеже]`:

```
Console.WriteLine(tuple.Item1); // 5
```

Существуют различные обобщенные классы `Tuple` для поддержки различного количества элементов от одного до восьми. В случае если имеется более восьми элементов, которые нужно включить в кортеж, можно использовать определение класса `Tuple` с восемью параметрами. Последний параметр называется `TRest`, в котором должен передаваться сам кортеж. Таким образом, есть возможность создавать кортежи с любым количеством параметров.

Также можно дать названия полям кортежа:

```
var tuple = (count:5, sum:10);
 Console.WriteLine(tuple.count); // 5

 Console.WriteLine(tuple.sum); // 10
```

Теперь, чтобы обратиться к полям кортежа, используются их имена, а не названия `Item1` и `Item2`.

Кортежи могут передаваться в качестве параметров в метод, могут быть возвращаемым результатом функции, либо использоваться иным образом.

Структуры

Структура – это более простая версия классов. Все структуры наследуются от базового класса `System.ValueType` и являются типами значений. Структуры могут содержать в себе обычные переменные и методы.

Структуры объявляются при помощи ключевого слова `struct`. Для примера создадим структуру `Book`, в которой будут храниться переменные для названия, автора и года издания книги. Кроме того, структура будет содержать метод для вывода информации о книге на консоль:

```
struct Book

{ public string name;
  public string author;
  public int year;

   //метод вывода информации
 public void Info()

{Console.WriteLine($"Книга'{name}'(автор{author})
была издана в {year} году");} }
```

Чтобы можно было использовать переменные и методы структуры, из любого места программы мы ставим перед переменными и методом модификатор доступа `public`. Структуру можно задать как внутри пространства имен, так и внутри класса, но не внутри метода. Экземпляр структуры можно создавать без ключевого слова `new`:

```
Book b;

 b. Name = "BookName";
```

По сути структура `Book` представляет новый тип данных. Мы также можем использовать массив структур:

```
Book[] books=new Book[3];
 books[0].name = "Война и мир";
 books[0].author = "Л. Н. Толстой";
 books[0].year = 1869;

 books[1].name = "Преступление и наказание";
 books[1].author = "Ф. М. Достоевский";
 books[1].year = 1866;
```

```
// вывод массива структур
foreach (Book b in books)
 {b.Info();}
```

Структуры подходят для создания несложных типов, таких как точка, цвет, окружность. Если необходимо создать множество экземпляров подобного типа, используя структуры, мы экономим память, которая могла бы выделяться под ссылки в случае с классами.

Конструкторы в структурах. Кроме обычных методов структура может содержать специальный метод – конструктор, который присваивает всем полям некоторые значения по умолчанию. Вызов конструктора имеет следующий синтаксис:

```
new название_структуры ([список_параметров])
```

Пример описания конструктора:

```
struct Book
  {  …
  // конструктор
    public Book(string n, string a, int y)
     { name = n;
       author = a;
       year = y;  }
     … }
```

Конструктор по сути является обычным методом, только не имеет возвращаемого значения, и его название всегда совпадает с именем структуры или класса. Теперь используем его:

```
Book book=new Book("Динка", "В.А. Осеева",1985);
 book.Info();
```

Теперь нам не надо вручную присваивать значения полям структуры – их инициализацию выполнил конструктор.

Практические задания

Задание 1. Разработать приложение с меню, состоящим из 4 пунктов, в виде методов с передачей параметров, реализовать передачу данных по значению, по ссылке, использовать выходные

параметры.

Задание 2. На 4-й пункт меню реализовать следующее задание: описать структуру (по вариантам). Создать массив структур из 5 элементов, заполнить их программно. Разработать метод, который будет возвращать полученный кортеж (по вариантам) с 5 элементами, по исходному кортежу с параметрами (по вариантам). Варианты даны в таблице.

Варианты заданий

Вариант	Структура	Исходный кортеж (входные параметры)	Полученный кортеж (выходные параметры)
1	2	3	4
1	**Магазин:** название, владелец, адрес, 3 оценки покупателя	название, адрес, дата открытия, оценка покупателя	название, владелец, код магазина, используя комбинацию владелец + дата открытия, средняя оценка покупателей, да или нет (определить, совпадает ли адрес).
2	**Театр:** название представления, время, тип места (партер, балкон), ряд, 3 цены билетов	название представления, № места, цена, время	название представления, тип места, номер билета, используя комбинацию тип места + ряд + № места, среднюю цену, да или нет (определить, совпадает ли время).
3	**Работники:** фамилия, название отдела, код должности, год поступления на работу, заработная плата за три месяца.	фамилия, номер трудового договора, заработная плата за последний месяц, год	фамилия, название отдела, номер пропуска, используя комбинацию код должности + номер трудового договора, среднюю заработную плату, да или нет (определить стаж работы > 5 лет).
4	**Магистрант:** фамилия, факультет, код специальности, оценки за три экзамена.	фамилия, номер группы, год поступления, оценка за последний экзамен	фамилия, факультет, номер зачетки, используя комбинацию код специальности и номер группы, средний балл за четыре экзамена, 1 или 2 (определить на каком курсе учится).
5	**Кинотеатр:** название фильма, время, тип фильма (3d или нет), ряд, цена билетов (3)	название фильма, место, цена, время	название фильма, тип места, номер билета, используя комбинацию тип фильма + ряд + место, среднюю цену, да или нет (определить, совпадает ли время).
6	**Анкета:** фами-	фамилия, год	фамилия, гражданство, пароль,

	лия, гражданство, адрес, 3 оценки.	рождения, оценка	используя комбинацию фамилия + год рождения, средняя оценка, количество полных лет.
7	**Фильмотека:** название фильма, жанр, страна, 3 цены	название фильма, год, страна, цена	название, жанр, код фильма, используя комбинацию жанр + год, средняя цена, да или нет (определить совпадает ли страна).
8	**Книжный:** название, количество страниц, автор, 3 цены	название, количество страниц, год, цена	название, автор, код книги, используя комбинацию автор + год, средняя цена, да или нет (определить, совпадает ли количество страниц).
9	**Регистрация:** фамилия, пол, адрес, 3 оценки.	фамилия, фамилия на английском, год рождения, оценка	фамилия, пол, email, используя комбинацию фамилия на английском + год рождения, средняя оценка, количество полных лет.
10	**Сериалы:** название, количество сезонов, страна, 3 оценки	название, год, количество сезонов, оценка	название, страна, код сериала, используя комбинацию страна + год, средняя оценка, да или нет (определить совпадает ли количество сезонов).
11	**Сувениры:** название, материал, страна, 3 цены	название, страна, дата изготовления, цена	Название, материал, код сувенира, используя комбинацию страна + дата, средняя цена, да или нет (определить, совпадает ли страна).
12	**Цветы:** название, вид, страна, наличие горшка, 3 цены	название, страна, дата, цена	Название, вид, код цветка, используя комбинацию вид + дата, средняя цена, да или нет (определить, совпадает ли страна).
13	**Мультфильмы:** название, продолжительность, страна, 3 цены	название, год, продолжительность, цена	название, страна, код мультфильма, используя комбинацию страна + год, средняя цена, да или нет (определить, совпадает ли продолжительность).
14	**Посуда:** название, материал, тип, 3 цены	название, тип, дата, цена	Название, материал, код посуды, используя комбинацию материал + дата, средняя цена, да или нет (определить, совпадает ли тип).
15	**Статьи:** название, количество страниц, журнал, 3 оценки	название, количество страниц, год, оценка	название, журнал, код статьи, используя комбинацию журнал + год, средняя оценка, да или нет (определить, совпадает ли количество страниц).
16	**Авторизация:** фамилия, пол, год рождения, 3 оценки.	фамилия, фамилия на английском, оценка	фамилия, пол, пароль, используя комбинацию фамилия на английском + год рождения, средняя оценка, количество полных лет.
17	**Телевизоры:** название, разрешение, страна, гарантия, 3 цены	название, страна, дата поставки, цена	название, разрешение, код телевизора, используя комбинацию страна + дата, средняя цена, да или нет (определить, совпадает ли страна).

18	**Журнал:** название, количество страниц, издательство, 3 цены	название, количество страниц, номер, цена	название, автор, код журнала, используя комбинацию издательство + номер, средняя цена, да или нет (определить, совпадает ли количество страниц).
19	**Альбомы:** название, жанр, количество дорожек, 3 цены	название, количество дорожек, автор, цена	название, жанр, код альбома, используя комбинацию жанр + автор, средняя цена, да или нет (определить, совпадает ли количество дорожек).
20	**Поезда:** станция прибытия, станция отбытия, машинист, продолжительность поездки, 3 цены	станция прибытия, продолжительность, время отправки, цена	станция прибытия, станция отбытия, номер поезда, используя комбинацию машинист + время отправки, средняя цена, да или нет (определить, совпадает ли продолжительность).
21	**Мебель:** название, материал, тип, 3 цены	название, тип, дата изготовления, цена	название, материал, код мебели, используя комбинацию материал + дата, средняя цена, да или нет (определить, совпадает ли тип).
22	**Ткани:** название, материал, цвет, ширина, 3 цены	название, цвет, количество рулонов, цена	название, материал, код ткани, используя комбинацию цвет + количество рулонов, средняя цена, да или нет (определить, совпадает ли цвет).
23	**Аэрофлот:** пункт назначения, тип самолета, количество мест, 3 цены	пункт назначения, количество мест, время отправки, цена	пункт назначения, тип самолета, номер рейса, используя комбинацию пункт назначения + время отправки, средняя цена, да или нет (определить, совпадает ли количество мест).
24	**Игрушки:** название, материал, тип, 3 цены	название, тип, для какого возраста, цена	название, материал, код игрушки, используя комбинацию тип + для какого возраста, средняя цена, да или нет (определить, совпадает ли тип).
25	**Инструменты:** название, материал, вид, 3 цены	название, вид, количество, цена	название, материал, код инструмента, используя комбинацию вид + количество, средняя цена, да или нет (определить, совпадает ли вид).
26	**Самолеты:** пункт назначения, тип самолета, количество мест, 3 цены	пункт назначения, количество мест, пилот, цена	пункт назначения, тип самолета, номер самолета, используя комбинацию тип самолета + количество мест, средняя цена, да или нет (определить, совпадает ли количество мест).

Окончание таблицы 8

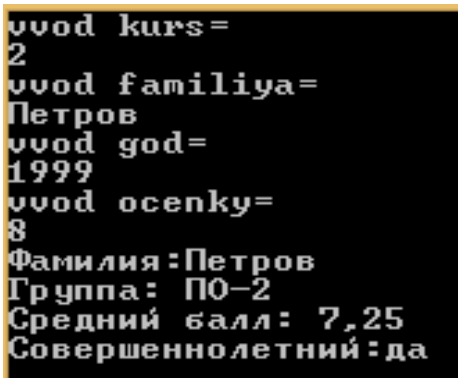
1	2	3	4
27	**Порт:** пункт назначения, тип судна, количество мест, 3 цены	пункт назначения, количество мест, время отправки, цена	пункт назначения, тип судна, номер судна, используя комбинацию пункт назначения + время отправки, средняя цена, да или нет (определить, совпадает ли количество мест).
28	**Лекарства:** название, назначение, тип, 3 цены	название, тип, для какого возраста, цена	название, назначение, код лекарства, используя комбинацию тип + для какого возраста, средняя цена, да или нет (определить, совпадает ли тип).
29	**Семена:** название, страна, тип, 3 цены	название, тип, количество, цена	название, страна, код семян, используя комбинацию тип + количество, средняя цена, да или нет (определить, совпадает ли тип).
30	**Техника:** название, страна, тип, 3 цены	название, тип, срок эксплуатации, цена	название, страна, код техники, используя комбинацию тип + срок эксплуатации, средняя цена, да или нет (определить, совпадает ли тип).

Пример выполнения задания 2. Создать структуру «Студент» с полями: фамилия, факультет, специализация, оценки за три экзамена. Создать массив структур из 2 элементов, заполнить их программно. Раз- работать метод, который будет возвращать кортеж с 5 элементами, по кортежу с параметрами: фамилия, курс, год рождения, оценка за послед- ний экзамен. Нужно вывести фамилию, группу, используя комбинацию: специальность и курс, средний балл за четыре экзамена (Результаты представлены на рисунке).

```
using System;
using
System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace ConsoleApp1
{class Program
{struct stud
```

```
{ public string fio;
  public string
  fakultet; public
  string spec; public
  int o1, o2, o3;

    }
static  Tuple<string,  string,  float,  string>  Cor-
teg(string fam1, int kurs1,int god1, int o4 )

 {string p = fam1,grup1,otv;
  float sum=0, sr;

  stud[] st = new stud[3];
  st[0].fio = "Петров";
  st[0].fakultet =
  "Математический"; st[0].spec =
  "ПО";

  st[0].o1 = 4;  st[0].o2 = 8;  st[0].o3 = 9;

  st[1].fio = "Сидоров";
  st[1].fakultet = "Биологический";
  st[1].spec = "Б";

  st[1].o1 = 10;  st[1].o2 = 7;  st[1].o3 = 8;

int nom = -1;

for (int i = 0; i <= 2;
 i++) if (st[i].fio ==
 fam1)

  { nom = i;
    break;
    }

if (nom != -1)

 {grup1 = st[nom].spec+"-" + Convert.ToString(kurs1);
  sum = st[nom].o1 + st[nom].o2 + st[nom].o3 + o4;

  sr = sum / 4;  }

 else { grup1 = "не определена"; sr = o4/4f;
} if (2017 - god1 >= 18) otv = "да";

 else otv = "нет";
```

```
    return Tuple.Create<string, string, float, string>(p,
    grup1, sr, otv);  }

    static void Main(string[] args)

     {int c;
      string
      text;

      Console.WriteLine("vvod kurs=");

      int kurs=Convert.ToInt16(Console.ReadLine());
      Console.WriteLine("vvod familiya=");

      string fam = Console.ReadLine();
      Console.WriteLine("vvod god=");

      int god = Convert.ToInt16(Console.ReadLine());
      Console.WriteLine("vvod ocenky=");

      int ocen = Convert.ToInt16(Console.ReadLine());
      var myTuple = Corteg(fam,kurs,god,ocen);

    Console.WriteLine("Фамилия:{0}\n"+          "Группа:

  {1}\n"+"Средний  балл:{2}\n"+"Совершеннолетний:{3}\n",

myTuple.Item1,myTuple.Item2,myTuple.Item3,my-
Tuple.Item4);

Console.ReadLine();  }}}
```



Результат выполнения программы