РЕАЛИЗАЦИЯ АЛГОРИТМОВ

Рекуррентные соотношения

Пусть $a_1, a_2, \dots, a_n$ – произвольная числовая последовательность. Рекуррентным соотношением называется такое соотношение между членами последовательности, в котором каждый следующий член выражается через несколько предыдущих, т.е.:

$$a_k = f(a_{k-1}, a_{k-2}, \dots, a_{k-l}), k > l \quad \textbf{(1)}$$

Последовательность задана рекуррентно, если для нее определено рекуррентное соотношение вида (1) и заданы первые l ее членов.

Самым простым примером рекуррентной последовательности является арифметическая прогрессия. Рекуррентное соотношение для нее записывается в виде: $a_k = a_{k-1} + d$, где d – разность прогрессии. Зная первый элемент и разность прогрессии, и, используя данное рекуррентное соотношение, можно последовательно вычислить все остальные члены прогрессии.

Пример 1

Рассмотрим пример программы, в которой вычисляются первые n членов арифметической прогрессии при условии, что $a_1 = \frac{1}{2}$ и $d = \frac{1}{4}$.

```
static void Main()
{
   double a =0.5;
   const double d = 0.25;
   Console.Write("n=");
   int n = int.Parse(Console.ReadLine());

   //вывели первый член последовательности
   Console.WriteLine("a1={0}", a);

   //организуем вычисление 2, 3, … ,n члена последовательности
   for (int i = 2; i <= n; i++)
   {
      //для этого прибавляем к предыдущему члену значение d
      a += d;
      //и выводим новое значение a на экран
      Console.WriteLine("a{0}={1}", i, a);
   }
}
```

Результат работы программы:

```
n    состояние экрана
5      a1: 0.5
       a2: 0.75
       a3: 1
       a4: 1.25
       a5: 1.5
```

Задание

Измените программу так, чтобы на экран выводился только n-ный член последовательности.

Пример 2

Другим примером рекуррентной последовательности является геометрическая прогрессия. Рекуррентное соотношение для нее записывается в виде:

$$b_k = b_{k-1} * q\ ,$$

где q – знаменатель прогрессии. Рассмотрим пример программы, в которой вычисляются первые n членов арифметической прогрессии при условии, что $b_1=1$, $q=2$.

```
static void Main()
{
   ulong b =1;
   const byte q = 2;
   Console.Write("n=");
   int n = int.Parse(Console.ReadLine());

   // вывели первый член последовательности
   Console.WriteLine("b1={0}", b);

   // организуем вычисление 2, 3, … ,n члена последовательности
   for (int i = 2; i <= n; i++)
   {
      // для этого прибавляем к предыдущему члену значение q
      b *= q;
      // и выводим новое значение b на экран
      Console.WriteLine("b{0}={1}", i, b);
   }
}
```

Результат работы программы:

```
n       состояние экрана
5          b1: 1
           b2: 2
           b3: 4
           b4: 8
           b5: 16
```


Задание

Измените программу так, чтобы на экран члены последовательности выводились в строчку через пробел (без указания номера).

Пример 3

В арифметической и в геометрической прогрессиях каждый член последовательности зависит только от одного предыдущего значения. Более сложная зависимость представлена в последовательности Фибоначчи: $a_1 = a_2 = 1$, $a_n = a_{n-1} + a_{n-2}$. В этом случае каждый член последовательности зависит от значений двух предыдущих членов. Рассмотрим пример программы, в которой вычисляются первые n членов последовательности Фибоначчи.

```
static void Main()
{
   int a1=1, a2=1, a3;//задаем известные члены последовательности
   Console.Write("n=");
   int n = int.Parse(Console.ReadLine());
   //выводим известные члены последовательности
   Console.WriteLine("a1={0}\na2={1}",a1,a2);

/*Организуем цикл для вычисления членов последовательности с
номерами 3, 4,…, n. При этом в переменной a1 будет храниться
значение члена последовательности с номером i-2, в переменной a2 -
члена с номером i-1; переменная a будет использоваться для
вычисления члена с номером i. */
   for (int i = 3; i <= n; i++)
   {
      // по рекуррентному соотношению вычисляем
      // член последовательности с номером i и выводим
      // его значение на экран
      a3=a1+a2;
      Console.WriteLine("a{0}={1}", i, a3);

      // выполняем рекуррентный пересчет для следующего шага цикла
      // в элемент с номером i-2 записываем значение
      // элемента с номером i-1
      a1 = a2;

      // в элемент с номером i-1 записываем значение
      // элемента с номером i
      a2 = a3;
   }
}
```

Результат работы программы:

```
n   состояние экрана
5      a1: 1
       a2: 1
       a3: 2
       a4: 3
       a5: 5
```

Задание

Измените программу так, чтобы на экран выводились только четные члены последовательности Фибоначчи.

Вычисление конечных сумм и произведений

Решение многих задач связано с нахождением суммы или произведения элементов заданной последовательности. В данном разделе мы рассмотрим основные приемы вычисления конечных сумм и произведений.

Пусть $u_1(x), u_2(x), \ldots, u_n(x)$ – произвольная последовательность n функций. Будем рассматривать конечную сумму вида $u_1(x) + u_2(x) + \ldots + u_n(x)$. Такую сумму можно записать более компактно, используя следующее обозначение:

$$u_1(x)+u_2(x)+\ldots+u_n(x)=\sum_{i=1}^{n}u_i(x)$$

При $n \le 0$ значение суммы равно 0.

В дальнейшем будем также использовать сокращенную запись для конечного произведения данной последовательности, которая выглядит следующим образом:

$$u_1(x)\cdot u_2(x)\cdot\ldots\cdot u_n(x)=\prod_{i=1}^{n}u_i(x)$$

Пример 1

Написать программу, которая подсчитывает сумму натуральных чисел от 1 до n (n≥1).

Указания по решению задачи. Пусть s_n – сумма натуральных чисел от 1 до n. Тогда s_n=1+2+…+(n-1)+n=(1+2+…+(n-1))+n=s_{n-1}+n, s_0=0. Мы пришли к рекуррентному соотношению s_0=0, s_n=s_{n-1}+n, которым мы можем воспользоваться для подсчета суммы. Соотношение s_n=s_{n-1}+n говорит о том, что сумма на n-ном шаге равна сумме, полученной на предыдущем шаге, плюс очередное слагаемое.

```
static void Main()
{
   Console.Write("Введите значение n: ");
   uint n = uint.Parse(Console.ReadLine());
   uint s = 0;
   for (uint i = 1; i <= n; i++)
   {
      s+=i;
   }
   Console.WriteLine("s="+s);
}
```

Результат работы программы:

```
n    s
5    15
481  115921
```

Задание

Измените программу так, чтобы на экран выводилось среднее арифметическое натуральных чисел от 1 до n.

Пример 2

Написать программу, которая подсчитывает n! для заданного натурального n.

Указание по решению задачи. Из свойства факториала:

0!=1!=1, n!=1*2*3*…*n, n!=(n-1)!n.

Следствовательно, факториал можно вычислять, используя рекуррентное соотношение

b_0=1, b_n=b_{n-1}*n.

Текст программы:

```
static void Main()
{
   Console.Write("Введите значение n: ");
   ulong n=ulong.Parse(Console.ReadLine());
   ulong f=1;
   for (ulong i=1; i<=n; ++i)
   {
      f*=i;
   }
   Console.WriteLine("f= "+f);
}
```

Результат работы программы:

```
n     f
5     120
20    2432902008176640000
```

Задание

Измените программу так, чтобы для заданного значения n *на экран выводились все вычисленные факториалы. Например, для* n=3 *на экран следует вывести:*

1!=1
2!=2
3!=6

Пример 3

Написать программу для подсчета суммы

$$S_n = \frac{\cos x}{1} + \frac{\cos x + \cos 2x}{2} + \frac{\cos x + cox2x + \cos 3x}{3} + \ldots + \frac{\cos x + \ldots + \cos nx}{n}$$

где x – вещественное число, n – натуральное число.

Указания по решению задачи. Если пронумеровать слагаемые, начиная с 1, то мы увидим, что номер слагаемого совпадает со значением знаменателя. Рассмотрим каждый числитель отдельно: $b_1 = \cos x$, $b_2 = \cos x + \cos 2x$, $b_3 = \cos x + \cos 2x + \cos 3x \ldots$ Эту последовательность можно представить рекуррентным соотношением:

$$b_0=0,\ b_n=b_{n-1}+\cos(nx) \qquad \textbf{(1)}$$

Теперь сумму можно представить следующим образом:

$$S_n = \frac{b_1}{1} + \frac{b_2}{2} + \frac{b_3}{3} + \ldots + \frac{b_n}{n}$$

для этой формулы справедливо рекуррентное соотношение:

$$S_0=0,\ S_n = S_{n-1} + \frac{b_n}{n} \qquad \textbf{(2)}$$

При составлении программы будем использовать формулы (1-2).

```
static void Main()
{
   Console.Write("Введите значение n: ");
   byte n = byte.Parse(Console.ReadLine());
   Console.Write("Введите значение x: ");
```

```
        double x = double.Parse(Console.ReadLine());
        double b = 0, s = 0;
        for (byte i = 1; i <= n; i++)
        {
           b += Math.Cos(i*x);
           s += b/i;
        }
        Console.WriteLine("s={0:f2}",s);
    }
```

Результат работы программы:

```
n    x    s
10   1    -0,46
120  -0,5 0,28
```

Задание

Измените программу так, чтобы вычислялось значение выражения:

$$S_n = -\frac{\cos x}{1} + \frac{\cos x + \cos 2x}{2} - \frac{\cos x + cox 2x + \cos 3x}{3} + \ldots (-1)^n \frac{\cos x + \ldots + \cos nx}{n}$$

Пример 4

Написать программу для подсчета суммы

$$S_n = \sum_{i=1}^{n} \frac{(-1)^{i+1} x^i}{i!}$$

где x – вещественное число, n – натуральное число.

Указания по решению задачи. Переходя от сокращенной формы записи к развернутой, получим:

$$S_n = \frac{x}{1!} - \frac{x^2}{2!} + \frac{x^3}{3!} - \ldots + \frac{(-1)^{n+1} x^n}{n!}$$

Каждое слагаемое формируется по формуле:

$$a_n = \frac{(-1)^{n+1} x^n}{n!}$$

Если в эту формулу подставить n=0, то получим:

$$a_0 = \frac{(-1)^1 x^0}{0!} = -1$$

Чтобы не вводить несколько рекуррентных соотношений (отдельно для числителя, отдельно для знаменателя), выразим последовательность слагаемых рекуррентным соотношением вида $a_n = a_{n-1} q$, где q для нас пока не известно. Найти его можно из выражения:

$$q = \frac{a_n}{a_{n-1}}$$

Произведя расчеты, получим, что

$$q = -\frac{x}{i}$$

Следовательно, для последовательности слагаемых мы имеем рекуррентное соотношение:

$$a_0 = -1,\ a_i = -a_{i-1} \cdot \frac{x}{i} \qquad \textbf{(3)}$$

а всю сумму, по аналогии с предыдущими примерами, можно представить рекуррентным соотношением:

$$S_0 = 0, \quad S_n = S_{n-1} + a_n \qquad \textbf{(4)}$$

Текст программы, решающей поставленную задачу с использованием формул (3) и (4), приведен ниже.

```
static void Main()
{
   Console.Write("Введите значение n: ");
   byte n = byte.Parse(Console.ReadLine());
   Console.Write("Введите значение x: ");
   double x = double.Parse(Console.ReadLine());
   double a = -1, s = 0;
   for (byte i = 1; i <= n; i++)
   {
      a *= -x/i;
      s += a;
   }
   Console.WriteLine("s={0:f2}",s);
}
```

Результат работы программы:

```
n     x     s
10    1     0,63
120   -0,5  -0,65
```

Задание

Измените программу так, чтобы вычислялось значение выражения:

$$S_n = \sum_{i=1}^{n} \frac{(-1)^{i+1} x^i}{(2i)!}$$

Пример 5

Написать программу для подсчета произведения:

$$P_k = \prod_{n=1}^{k} (1 + \frac{x^{2n} + x^n}{n})$$

где x – вещественное число, n – натуральное число.

Указания по решению задачи. Преобразуем заданное выражение к виду:

$$P_k = \prod_{n=1}^{k} (1 + \frac{x^n (x^n + 1)}{n})$$

и перейдем от сокращенной формы записи к развернутой:

$$P_k = (1 + \frac{x^1 (x^1 + 1)}{1})(1 + \frac{x^2 (x^2 + 1)}{2})...(1 + \frac{x^k (x^k + 1)}{k})$$

В числителе каждой дроби встречается x^n (см. пример 2), его можно вычислить по рекуррентному соотношению:

$$b_0=1,\ b_n=b_{n-1}*x \qquad \textbf{(5)}$$

Тогда произведение можно представить как:

$$P_k=(1+\frac{b_1(b_1+1)}{1})(1+\frac{b_2(b_2+1)}{2})...(1+\frac{b_k(b_k+1)}{k})$$

что, в свою очередь, можно выразить рекуррентным соотношением:

$$P_0=1,\ P_k=P_{k-1}*(1+\frac{b_k(b_k+1)}{k}) \qquad \textbf{(6)}$$

При составлении программы будем пользоваться формулами (5-6).

```
static void Main()
{
   Console.Write("Введите значение n: ");
   byte n = byte.Parse(Console.ReadLine());
   Console.Write("Введите значение x: ");
   double x = double.Parse(Console.ReadLine());
   double b = 1, p = 1;
   for (byte i = 1; i <= n; i++)
   {
      b *= x;
      p *= (1+b*(b+1)/i);
   }
   Console.WriteLine("p={0:f2}",p);
}
```

Результат работы программы:

```
n     x     s
10    1     66,00
20    -0,5  0,85
```

Задание

Измените программу так, чтобы вычислялось значение выражения:

$$P_k=\prod_{n=1}^{k}(1+\frac{x^{2n}+x^n}{n!})$$

Вычисление бесконечных сумм

Будем теперь рассматривать бесконечную сумму вида:

$$u_1(x)+u_2(x)+\ldots+u_n(x)+\ldots=\sum_{i=1}^{\infty}u_i(x)$$

Это выражение называется функциональным рядом. При различных значениях *x* из функционального ряда получаются различные числовые ряды:

$$a_1+a_2+\ldots+a_n+\ldots=\sum_{i=1}^{\infty}a_i$$

Числовой ряд может быть сходящимся, или расходящимся. Совокупность значений *x*, при которой функциональный ряд сходится, называется его областью сходимости.

Числовой ряд называется сходящимся, если сумма n первых его членов $S_n = a_1 + a_2 + \ldots + a_n$ при $n \to \infty$ имеет предел; в противном случае, ряд называется расходящимся. Ряд может сходиться лишь при условии, что общий член ряда a_n при неограниченном увеличении его номера стремится к нулю:

$$\lim_{n\to\infty} a_n = 0$$

Это необходимый признак сходимости для всякого ряда.

В случае бесконечной суммы будем вычислять ее с заданной точностью *e*. Считается, что требуемая точность достигается, если вычислена сумма нескольких первых слагаемых и очередное слагаемое оказалось по модулю меньше, чем *e*.

Пример 1

Написать программу для подсчета суммы:

$$\sum_{i=1}^{\infty} \frac{(-1)^i}{i!}$$

с заданной точностью e (e>0).

Указание по решению задачи. Рассмотрим, что представляет собой заданный ряд:

$$\sum_{i=1}^{\infty} \frac{(-1)^i}{i!} = -\frac{1}{1} + \frac{1}{2} - \frac{1}{6} + \frac{1}{24} - \frac{1}{120} + \ldots + \frac{1}{\infty}$$

Общий член ряда с увеличением значения i стремится к нулю, следовательно, данную сумму будем вычислять с определенной точностью *e*. Заметим также, что последовательность слагаемых можно выразить с помощью рекуррентного соотношения:

$$a_1 = -1,\ a_i = \frac{-a_{i-1}}{i},$$

а всю сумму - с помощью рекуррентного соотношения $S_0=0$, $S_n=S_{n-1}+a_n$. (выведите данные рекуррентные соотношения самостоятельно.)

```
static void Main()
{
   Console.Write("Задайте точность вычислений e: ");
   double e = double.Parse(Console.ReadLine());
   double a = -1, s = 0;
   for (int i = 2; Math.Abs(a) >= e; i++)
   {
      s += a;
      a /= -i;
   }
   Console.WriteLine("s={0:f8}", s);
}
```

Результат работы программы:

```
e              s
0,1       -0,66666667
0,01      -0,62500000
0,001     -0,63194444
0,0001    -0,63214286
```

Задание 1

Объясните, почему при разных значениях точности мы получили разные значения суммы.

Задание 2

Измените программу так, чтобы на экран выводилось не только значение суммы, но и количество слагаемых.

Пример 2

Вычислить значение функции:

$$F(\mathrm{x}) = -\frac{1}{(x+1)} + \frac{(x-1)^2}{2(x+1)^2} - \frac{(x-1)^4}{4(x+1)^3} + \frac{(x-1)^6}{8(x+1)^4} - \ldots$$

на отрезке *[a,b]* с шагом *h* и точностью ε. Результат работы программы представить в виде таблицы, которая содержит номер аргумента, значение аргумента, значение функции.

Указания по решению задачи. Перейдем от развернутой формы записи функции к сокращенной, получим:

$$F(x) = \sum_{i=1}^{\infty} \frac{(-1)^i (x-1)^{2i-2}}{2^{i-1}(x+1)^i}$$

и воспользуемся приемом, рассмотренным в предыдущем разделе. Получим, что слагаемые данной функции определяются с помощью рекуррентного соотношения:

$$c_1 = -\frac{1}{(x+1)}, \; c_i = \frac{-b_{i-1}(x-1)^2}{2(x+1)}.$$

Текст программы:

```
static void Main()
{
  Console.Write("a: ");
  double a = double.Parse(Console.ReadLine());
  Console.Write("b: ");
  double b = double.Parse(Console.ReadLine());
  Console.Write("h: ");
  double h = double.Parse(Console.ReadLine());
  Console.Write("e: ");
  double e = double.Parse(Console.ReadLine());
  double x;
  byte i;
  //выводим заголовок таблицы
  Console.WriteLine("{0,2} {1,6} {2,10}",'#', 'x', 's');
```

```
   // строим таблицу на отрезке [a, b] с шагом h
   for (x = a, i = 1; x <= b; x += h, i++)
   {
      // определяем начальное значение суммы и
      // первое слагаемое для заданного x
      double c = 1/(x+1);
      double s = 0;

      //пока не достигнута заданная степень точности
      while(Math.Abs(c) >= e)
      {
         //добавляем слагаемое к сумме
         s += c;
         //формируем очередное слагаемое
         c *= -Math.Pow(x-1,2)/(2*(x+1));
      }
      //выводим полученные данные
      Console.WriteLine("{0,2} {1,6:f2} {2,10:f4}",i, x, s);
   }
}
```

Результат работы программы:

```
i   x     s
1   1,00  -0.5000
2   1,30  -0.4264
3   1,60  -0.3597
4   1,90  -0.3026
```

Задание

Измените программу так, чтобы на экран выводилось таблица с заголовком «i   x   s   n», где n это количество слагаемых для соответствующего значения x.

Алгоритмы поиска делителей натурального числа

По определению, целое число i является делителем натурального числа N, если при делении N на i остаток от деления равен нулю. Поэтому, чтобы найти все делители числа N, нужно перебрать все натуральные числа от 1 до N, и проверить, являются ли они его делителями. Данный алгоритм можно реализовать с помощью следующей программы:

```
static void Main()
{
   Console.Write("n: ");
   uint n = uint.Parse(Console.ReadLine());
   for (uint i = 1; i <= n; i++)
   {
      if (n%i == 0)
      {
         Console.Write(“{0} ”, i);
      }
   }
}
```

Результат работы программы:

```
n       Сообщение на экране
100     1 2 4 5 10 20 25 50 100
```

Задание

Измените программу так, чтобы на экран выводились только делители, являющиеся нечетными числами.

Рассмотренный алгоритм решает поставленную задачу, выполняя 100 итераций (повторений) цикла при N=100. Данный алгоритм работает неэффективно, т.к. для нахождения делителей перебираются все числа от 1 до N. Однако для любого числа 1 и само число являются его тривиальными делителями. Поэтому из диапазона следует исключить числа 1, N. Более того, наибольшим делителем, отличным от самого числа N, может быть N/2, а все числа большие N/2 заведомо не могут быть его делителями. Поэтому из рассматриваемого диапазона нужно исключить все числа большие N/2. Тогда для поиска делителей числа N можно перебирать все натуральные числа из диапазона от 2 до N / 2. В результате, диапазон исследуемых чисел сократился в два раза, что для больших значений N дает выигрыш во времени выполнения программы. Усовершенствованный алгоритм можно реализовать следующей программой:

```
static void Main()
{
   Console.Write("n: ");
   uint n=uint.Parse(Console.ReadLine());
   Console.Write("1 ");
   for (uint i = 2; i <= n/2; i++)
   {
      if (n%i == 0)
      {
         Console.Write(“{0} ”, i);
      }
   }
   Console.WriteLine(n);
}
```


Результат работы программы:

```
n      Сообщение на экране
100    1 2 4 5 10 20 25 50 100
```

Задание

Измените программу так, чтобы на экран выводились только делители, являющиеся нечетными числами.

Рассмотренный алгоритм решает поставленную задачу, выполняя 50 итераций цикла при N=100. Однако и данный алгоритм можно усовершенствовать, если вспомнить тот факт, что если i является делителем числа N, то и число N/i также будет являться его делителем. Например, если число 100 делится на 2, то оно делится и на 50 (т.е., 100/2). Таким образом, почти все делители образуют пару, и если мы нашли один делитель, то можем определить и парный ему. Исключение составляют только такие делители, квадраты которых равны самому числу. Например, число 10 является делителем числа 100, но т.к. 10^2=100, то у этого делителя числа 100 нет парного. Для таких делителей выполняется свойство i=$\sqrt{N}$, поэтому для поиска делителей числа N можно перебирать все натуральные числа из диапазона от 1 до $\sqrt{N}$. Усовершенствованный алгоритм можно реализовать следующей программой:

```
static void Main()
{
   Console.Write("n: ");
   uint n = uint.Parse(Console.ReadLine());
   for (uint i = 1; i <= Math.Sqrt(n); i++)
   {
      if (n%i == 0) //если i делитель n
      {
         if (i*i == n) //и i нет парного делителя
         {
            Console.Write("{0} ", i);
         }
         else
         {
            // иначе выводим i и его парный делитель
            Console.Write("{0} {1} ",i,n/i);
         }
      }
   }
}
```

Результат работы программы:

```
n     Сообщение на экране
100   1 100 2 50 4 25 5 20 10
```

Задание

Измените программу так, чтобы на экран выводились делители, оканчивающиеся на нечетную цифру.

Рассмотренный алгоритм решает поставленную задачу, выполняя 10 итераций цикла при N=100. Таким образом, данный алгоритм будет работать быстрее, чем предыдущие алгоритмы.

Алгоритм поиска всех делителей заданного натурального числа имеет много приложений. Например, с его помощью можно установить, является ли заданное натуральное число простым или составным. Напомним, что число, которое делится только на 1 и само на себя, называется простым; все остальные натуральные числа называются составными. Таким образом, с помощью алгоритма поиска делителей можно осуществить поиск нетривиальных делителей, а затем провести проверку: если найдется хотя бы один нетривиальный делитель, то число составное, иначе – простое. Данный алгоритм можно реализовать с помощью следующей программы:

```
static void Main()
{
   Console.Write("n (n>1): ");
   uint n = uint.Parse(Console.ReadLine());
   byte k = 0; // количество нетривиальных делителей
```

```
        for (uint i = 2; i <= Math.Sqrt(n); i++)
        {
           if (n%i == 0)
           {
              ++k;
              break;
           }
        }
        if (k==0)
        {
           Console.WriteLine("Число простое");
        }
        else
        {
           Console.WriteLine("Число составное");
        }
     }
```

Результат работы программы:

```
N      Сообщение на экране
8      число составное
111    число простое
```

С помощью предложенного алгоритма можно решить и более сложную задачу. Например, дано число N. Составить программу вывода следующего за ним простого числа K. Идея решения данной задачи следующая: перебираем все натуральные числа, начиная с N + 1 с шагом 1 до тех пор, пока не встретим простое число. Для решения задачи используем два цикла. Первый, внешний, перебирает натуральные числа, начиная с N+1, второй, внутренний, определяет, является ли рассматриваемое число простым, или нет. Приведенный алгоритм можно реализовать следующей программой:

```
static void Main()
{
   Console.Write("n: ");
   uint n = uint.Parse(Console.ReadLine());
   byte k;
   do //внешний цикл
   {
      ++n; // берем следующее натуральное число
      k=0;
      //подсчитываем для него количество делителей
      for (uint i = 1; i <= Math.Sqrt(n); i++)
      {
         if (n%i == 0)
         {
            if (i*i == n)
            {
               ++k;
            }
            else
            {
               k+=2;
            }
```

```
          }
        }
      }
      while (k!=2); // останавливаем перебор чисел тогда,
                    // когда встретим простое число
      Console.WriteLine(n);
    }
```

Результат работы программы:

```
n    Сообщение на экране
5    7
74   79
```

Задание

Измените программу так, чтобы для заданного натурального числа n *на экран выводилось предыдущее простое число.*

Алгоритм, раскладывающий натуральное число на цифры

Известно, что любое натуральное число $A = a_n a_{n-1} ... a_1 a_0$, где a_n, a_{n-1}, ..., a_0 – цифры числа, можно представить следующим образом:

$$A = a_n a_{n-1} ... a_1 a_0 = a_n \cdot 10^n + a_{n-1} \cdot 10^{n-1} + ... + a_1 \cdot 10^1 + a_0 \cdot 10^0 =$$
$$((...((a_n \cdot 10 + a_{n-1}) \cdot 10 + a_{n-2}) \cdot 10 ...) \cdot 10 + a_1) \cdot 10 + a_0.$$

Например, число 1234 можно представить как:

$$1234 = 1 \cdot 10^3 + 2 \cdot 10^2 + 3 \cdot 10^1 + 4 \cdot 10^0 = ((1 \cdot 10 + 2) \cdot 10 + 3) \cdot 10 + 4 .$$

Из данного представления видно, что получить последнюю цифру можно, если найти остаток от деления числа на 10. Поэтому, для разложения числа на составляющие его цифры можно использовать следующий алгоритм:

1) находим остаток от деления числа N на 10, т.е., получаем крайнюю правую цифру числа.
2) находим целую часть при делении N на 10, заменяем число N получившимся результатом (т.е., отделяем от числа N крайнюю правую цифру).
3) если N > 0, то переходим на пункт 1; иначе число равно нулю и отделять от него больше нечего.

Предложенный алгоритм можно реализовать следующей программой:

```
static void Main()
{
   Console.Write("n: ");
   uint n = uint.Parse(Console.ReadLine());
   uint b;
   while (n > 0)
   {
      b = n%10;
      Console.Write("{0} ", b);
      n /= 10;
   }
}
```

Результат работы программы:

```
n             Сообщение на экране
5             5
1234567890    0 9 8 7 6 5 4 3 2 1
```

Недостатком данного алгоритма является то, что цифры числа выводятся на экран в обратном прядке.

Задание

Измените программу так, чтобы на экран выводилась только старшая цифра заданного натурального числа n.

Данный алгоритм можно использовать для решения различных задач. Например, можно найти сумму цифр заданного натурального числа:

```
static void Main()
{
   Console.Write("n: ");
   uint n = uint.Parse(Console.ReadLine());
   uint s = 0;
   while (n > 0)
   {
      s += n%10;
      n /= 10;
   }
   Console.WriteLine("s= {0}", s);
}
```

Результат работы программы:

```
n           sum
0           0
1234567890  45
```

Задание

Измените программу так, чтобы на экран выводилось среднее арифметическое цифр заданного натурального числа n.

С помощью предложенного алгоритма можно решать и более сложные задачи. Например, для заданного натурального числа N можно найти ближайшее, меньшее данного число, сумма цифр которого кратна натуральному числу C.

```
static void Main()
{
   Console.Write("n: ");
   uint n = uint.Parse(Console.ReadLine());
   Console.Write("c: ");
   uint c = uint.Parse(Console.ReadLine());
   uint s;
   do
   {
      // берем предыдущее натуральное число и запоминаем его копию
      --n;
      uint a = n;
      // находим сумму цифр числа a
```

```
         s = 0;
         while (a>0)
         {
            s += a % 10;   a /= 10;
         }
      }
      while (s % c != 0); // останавливаем перебор тогда, когда
                          //сумма цифр a станет кратной c
      Console.WriteLine("n= {0}", n);
   }
```

Результат работы программы:

```
N    C    Сообщение на экран
65   15   0
325  15   294
```

Задание

Измените программу так, чтобы для заданного натурального числа n *на экран выводилось ближайшее, большее данного число, сумма цифр которого кратна числу С.*

Алгоритмы нахождения наибольшего общего делителя двух натуральных чисел

Пусть даны два натуральных числа А и В. Если и А, и В одновременно делятся на число С, причем С наибольшее из всех возможных делителей, то С называется наибольшим общим делителем, или НОД. Если НОД чисел А и В равен 1, то эти числа называются взаимнопростыми.

Для нахождения НОД двух натуральных чисел можно воспользоваться алгоритмом Евклида:

1) задать два числа;
2) пока числа не равны, заменять большее число разностью большего и меньшего;
3) вывести в качестве результата любое из чисел.

Данный алгоритм можно реализовать следующей программой:

```
static void Main()
{
   Console.Write("a: ");
   uint a = uint.Parse(Console.ReadLine());
   Console.Write("b: ");
   uint b = uint.Parse(Console.ReadLine());
   while (a != b)
   {
      if (a > b)
      {
         a -= b;
      }
      else
      {
         b -= a;
      }
   }
   Console.WriteLine("НОД= {0}", a);
}
```

Результат работы программы:

```
a      b      Ответ
128    160    32
325    111    1
```

Для нахождения НОД двух натуральных чисел можно воспользоваться модификацией алгоритма Евклида:

1) задать два числа;
2) пока оба числа не равны нулю, заменять большее число остатком от деления большего числа на меньшее число;
3) вывести в качестве результата сумму преобразованных чисел.

Данный алгоритм можно реализовать следующей программой:

```
static void Main()
{
   Console.Write("a: ");
   uint a = uint.Parse(Console.ReadLine());
   Console.Write("b: ");
   uint b = uint.Parse(Console.ReadLine());
   while (a > 0 && b > 0)
   {
      if (a > b)
      {
         a %= b;
      }
      else
      {
         b %= a;
      }
   }
   Console.WriteLine("нод= {0}", a + b);
}
```


Если воспользоваться свойством $НОК(A,B)=\frac{A\cdot B}{НОД(A,B)}$, то данный алгоритм также можно использовать для нахождения наименьшего общего кратного (НОК).

```
static void Main()
{
   Console.Write("a: ");
   uint a = uint.Parse(Console.ReadLine());
   Console.Write("b: ");
   uint b = uint.Parse(Console.ReadLine());
   // первоначально записываем в nok произведение чисел a и b
   uint nok = a*b;
   //вычисляем НОД
   while (a>0 && b>0)
   {
      if (a>b)
      {
         a %= b;
      }
```

```
            else
            {
               b %= a;
            }
         }
         uint nod = a + b;
         nok /= nod;     // вычисляем НОК
         Console.WriteLine("нoк= {0}", nok);
      }
```

Результат работы программы:

```
A    B     НОК
21   20    420
21   35    105
```

Задание

Объясните, какой из алгоритмов поиска НОД будет работать более эффективно и почему.

С помощью предложенного алгоритма можно решать и более сложные задачи. Например, проверим, будет ли НОД двух натуральных чисел простым числом.

```
      static void Main()
      {
         Console.Write("a: ");
         uint a = uint.Parse(Console.ReadLine());
         Console.Write("b: ");
         uint b = uint.Parse(Console.ReadLine());
         //вычисляем nod заданных чисел
         while (a>0 && b>0)
         {
            if (a>b)
            {
               a %= b;
            }
            else
            {
               b %= a;
            }
         }
         uint nod = a + b;
         // находим количесто делителей у nod
         byte k = 0;
         for (uint i = 1; i <= Math.Sqrt(nod); i++)
         {
            if (nod % i == 0)
            {
               if (i * i == nod)
               {
                  ++k;
               }
               else
               {
                  k += 2;
```

```
                }
            }
        }
        if (k == 2)
        {
            Console.WriteLine("Нод={0} является простым числом",nod);
        }
        else
        {
            Console.WriteLine("Нод={0} является составным числом",nod);
        }
    }
```

Результат работы программы:

```
A      B      Сообщение на экране
22     33     Нод = 11 является простым числом
128    160    Нод = 32 является составным числом
```

Практикум №4

Задание 1.

Написать программу, вычисляющую первые n элементов заданной последовательности:

1. $b_1 = 9, b_n = 0.1b_{n-1} + 10$;
2. $b_1 = -1, b_n = 9 - 2b_{n-1}$;
3. $b_1 = 1, b_n = 0.2b_{n-1}^4 + 1$;
4. $b_1 = 4.7, b_n = \sin(b_{n-1}) + \pi$;;
5. $b_1 = 0.1, b_n = \frac{1}{6}(0.05 + b_{n-1}^3)$;
6. $b_1 = 2, b_n = 0.5(\frac{1}{b_{n-1}} + b_{n-1})$
7. $b_1 = 5, \quad b_n = (-1)^n b_{n-1} - 8$;
8. $b_1 = -1, \quad b_2 = 1, \quad b_n = 3b_{n-1} - 2b_{n-2}$;
9. $b_1 = -10, \quad b_2 = 2, \quad b_n = |b_{n-2}| - 6b_{n-1}$;
10. $b_1 = 2, \quad b_2 = 4, \quad b_n = 6b_{n-1} - b_{n-2}$;
11. $b_1 = 5, \quad b_n = \frac{b_{n-1}}{n^2 + n + 1}$;
12. $b_1 = 0.5, \quad b_2 = 0.2, \quad b_{n+1} = b_n^2 + \frac{b_{n-1}}{n}$;
13. $b_1 = 1, \quad b_n = \frac{1}{4}\left(3b_{n-1} + \frac{1}{3b_{n-1}}\right)$;
14. $b_1 = 2, \quad b_2 = 1, \quad b_n = \frac{2}{3}b_{n-2} - \frac{1}{3}b_{n-1}^2$;
15. $b_1 = 1, \quad b_2 = 2, \quad b_n = \frac{b_{n-2}}{4} + \frac{5}{b_{n-1}^2}$;
16. $b_1 = 1, \quad b_2 = 2, \quad b_n = \frac{nb_{n-2} - b_{n-1}}{n+1}$;
17. $b_1 = 4, \quad b_2 = 2, \quad b_n = \frac{b_{n-2}}{n} + \frac{n^2}{b_{n-1}}$
18. $b_1 = 100, \quad b_{2n} = b_{2n-1}/10, \quad b_{2n+1} = b_{2n} + 10$;

19. $b_1 = 0, \quad b_{2n} = b_{2n-1} + 3, \quad b_{2n+1} = 2b_{2n};$

20. $b_1 = 1, \quad b_2 = 5 \quad b_{2n} = b_{2n-1} + b_{2n-2}, \quad b_{2n+1} = b_{2n} - b_{2n-1}.$

Задание 2

Для заданного натурального n и действительного x подсчитать следующие суммы:

1. $S = 1^2 + 2^2 + 3^2 + \ldots + n^2;$

2. $S = \sqrt{1} + \sqrt{2} + \sqrt{3+} \ldots + \sqrt{n};$

3. $S = 1 + \frac{1}{2} + \frac{1}{3} + \ldots + \frac{1}{n};$

4. $S = 1 + \frac{1}{2^2} + \frac{1}{3^2} + \ldots + \frac{1}{n^2};$

5. $S = 1 + \frac{1}{\sqrt{2}} + \frac{1}{\sqrt{3}} + \ldots + \frac{1}{\sqrt{n}};$

6. $S = \frac{1}{\sin 1} + \frac{1}{\sin 2} + \ldots + \frac{1}{\sin n};$

7. $S = 1+2+2^2+2^3+\ldots+2^n;$

8. $S = \cos 1 - \cos 2 + \cos 3 - \ldots + (-1)^{n+1} \cos n;$

9. $S = 1!+2!+3!+\ldots+n!;$

10. $S = 1 - 3 + 3^2 - 3^3 + \ldots(-1)^n 3^n;$

11. $S = 1! - 2! + 3! - \ldots + (-1)^{n+1} n!;$

12. $S = \sin x + \sin x^2 + \sin x^3 + \ldots + \sin x^n;$

13. $S = 1 + \frac{1}{2!} + \frac{1}{3!} + \ldots + \frac{1}{n!};$

14. $S = -\frac{1}{2} + \frac{1}{2^2} - \frac{1}{2^3} + \ldots + \frac{(-1)^n}{2^n};$

15. $S = 1^3 - 2^3 + 3^3 \ldots + (-1)^{n+1} n^3;$

16. $S = x + 3x^3 + 5x^5 + 7x^7 + \ldots + (2n-1)x^{2n-1};$

17. $S = \frac{\cos x}{1} + \frac{\cos^2 x}{2} + \frac{\cos^3 x}{3} + \ldots + \frac{\cos^n x}{n};$

18. $S = \frac{1}{3^2} - \frac{1}{5^2} + \frac{1}{7^2} - \ldots + \frac{(-1)^{n+1}}{(2n+1)^2}$

19 $S = \frac{1}{\sin 1} + \frac{1}{\sin 1 + \sin 2} + \ldots + \frac{1}{\sin 1 + \sin 2 + \ldots + \sin n};$

20. $S = \sin x + \sin \sin x + \sin \sin \sin x + \ldots + \underbrace{\sin \sin \sin \ldots \sin x}_{n \text{ раз}};$

Задание 3

Для заданного натурального k и действительного x подсчитать следующие выражения:

1. $S = \sum_{n=1}^{k} \frac{x^n}{n};$

2. $S = \sum_{n=1}^{k} \frac{2^n \cdot n!}{n^2}$

3. $S = \sum_{n=1}^{k} \frac{(-1)^{n+1}}{n^2};$

4. $S = \sum_{n=1}^{k} \frac{x^{2(n-1)}}{(2+4(n-1))^2};$

5. $S = \sum_{n=1}^{k} \frac{(-1)^{n+1} x^{2n-1}}{(2n-1)!};$

6. $S = \sum_{n=1}^{k} \frac{1}{n \cdot n!};$

7. $S = \sum_{n=0}^{k} \frac{(-1)^{n+1} x^{2n+1}}{(2n+1)!}$;

8. $S = \sum_{n=1}^{k} \frac{(-1)^{n-1} x^{n}}{(2n)!}$;

9. $S = \sum_{n=1}^{k} \frac{(-1)^{n-1} x^{2n}}{(2n)!}$;

10. $S = \sum_{n=1}^{k} \frac{(-1)^{n} x^{n}}{2^{n} 7n}$;

11. $P = \prod_{n=1}^{k} (1 + \frac{x^{n}}{n^{2}})$;

12. $P = \prod_{n=1}^{k} (1 + \frac{x^{2n+1}}{n(n+1)})$;

13. $P = \prod_{n=1}^{k} (1 - \frac{x^{n}}{n!})$

14. $P = \prod_{n=1}^{k} (1 + \frac{(-1)^{n} x^{2n}}{n^{3}})$;

15. $P = \prod_{n=1}^{k} (1 + \frac{(-1)^{n+1} x^{n}}{n!})$;

16. $P = \prod_{n=1}^{k} (1 + \frac{x^{2n}}{n(n+4)})$;

17. $P = \prod_{n=1}^{k} (1 + \frac{(-1)^{n} x^{2n+1}}{n^{3} + n^{2}})$;

18. $P = \prod_{n=1}^{k} (1 + \frac{x^{n}}{2n!})$;

19. $P = \prod_{n=2}^{k} (1 + \frac{(-1)^{n} x^{2n-1}}{n^{3} - 1})$;

20. $P = \prod_{n=0}^{k} (1 + \frac{(-1)^{n-1} x^{2n}}{(n+2)(n+1)})$;

Задание 4

Вычислить бесконечную сумму ряда с заданной точностью e (e>0).

1. $\sum_{i=1}^{\infty} \frac{1}{i^{2}}$

2. $\sum_{i=1}^{\infty} \frac{1}{(i+1)^{3}}$

3. $\sum_{i=2}^{\infty} \frac{(-1)^{i}}{i^{2} - 1}$

4. $\sum_{i=1}^{\infty} \frac{1}{i(i+1)}$

5. $\sum_{i=2}^{\infty} \frac{5}{(i+1)(i-1)}$

6. $\sum_{i=1}^{\infty} \frac{(-2)^{i+1}}{i(2i+1)}$

7. $\sum_{i=1}^{\infty} \frac{2}{i!}$

8. $\sum_{i=1}^{\infty} \frac{1}{(2i)!}$

9. $\sum \frac{(-1)^{i}}{(2i-1)!}$

10. $\sum_{i=1}^{\infty} \frac{(-1)^{i+1}}{2i!}$

11. $\sum_{i=1}^{\infty} \frac{(-1)^{2i}}{i(i+1)(i+2)}$

12. $\sum_{i=3}^{\infty} \frac{(-1)^{2i-1}}{i(i-1)(i-2)}$

13. $\sum_{i=1}^{\infty} \frac{(-3)^{2i}}{3i!}$

14. $\sum_{i=1}^{\infty} \frac{(-5)^{2i-1}}{5(2i-1)!}$

15. $\sum_{i=1}^{\infty} \frac{1}{2^{i}}$

16. $\sum_{i=1}^{\infty} \frac{1}{3^{i} + 4^{i}}$

17. $\sum_{i=1}^{\infty} \frac{1}{5^{i} + 4^{i+1}}$

18. $\sum_{i=1}^{\infty} \frac{(-1)^{i}}{2^{2i}}$

19. $\sum_{i=1}^{\infty} \frac{(-1)^{i+1}}{3^{2i-1}}$

20. $\sum_{i=1}^{\infty} \frac{1}{\sqrt{3^{i}}}$

Задание 5

Вычислить и вывести на экран значение функции F(x) на отрезке *[a,b]* с шагом *h=0.1* и точностью ε. Результат работы программы представить в виде следующей таблицы:

№	Значение x	Значение функции F(x)	Количество просуммированных слагаемых n
1			
2			
…			

1. F(x) = $1+\frac{x^2}{4}+\frac{x^3}{4^2}+\frac{x^4}{4^3}+\frac{x^5}{4^4}+\ldots,\quad x\in[0.1;\,0.9]$.

2. F(x)= $1-\frac{x}{2\cdot 7}+\frac{x^2}{4\cdot 14}-\frac{x^3}{8\cdot 21}+\frac{x^4}{16\cdot 28}-\ldots,\quad x\in[0;\,0.9]$.

3. F(x) = $1-\frac{x^2}{1\cdot 3\cdot 4}+\frac{x^4}{2\cdot 4\cdot 5}-\frac{x^6}{3\cdot 5\cdot 6}+\frac{x^8}{4\cdot 6\cdot 7}-\ldots$, $x\in[0.2;\,0.7]$.

4. F(x) = $1+\frac{x^3}{3\cdot 2}+\frac{x^5}{5\cdot 2^2}+\frac{x^7}{7\cdot 2^3}+\ldots,\quad x\in[0;\,0.99]$.

5. F(x) = $1+\frac{x}{1\cdot 4}-\frac{x^2}{2\cdot 5}+\frac{x^3}{3\cdot 6}-\frac{x^4}{4\cdot 7}+\ldots,\quad x\in[0.1;\,0.9]$.

6. F(x) = $1-\frac{x^3}{3\cdot 4^2}+\frac{x^5}{4\cdot 5^2}-\frac{x^7}{5\cdot 6^2}+\ldots,\quad x\in[0;\,0.8]$.

7. F(x) = $1+\frac{x}{3}+\left(\frac{x}{3}\right)^2+\ldots,\quad x\in[0;\,0.9]$.

8. F(x) = $1+\frac{x^2}{2\cdot 4}+\frac{x^3}{4\cdot 6}+\frac{x^4}{6\cdot 8}+\ldots,\quad x\in[0.2;\,0.6]$.

9. F(x) = $1+\frac{x}{1\cdot 3}+\frac{x^2}{1\cdot 3\cdot 5}+\frac{x^3}{1\cdot 3\cdot 5\cdot 7}+\ldots,\quad x\in[0.05;\,0.95]$.

10. F(x)= $-\frac{\pi}{2}-\frac{1}{x}+\frac{1}{3x^3}-\frac{1}{5x^5}+\frac{1}{7x^7}-\ldots.,\quad x\in[-3;\,-2]$.

11. F(x)=$1+x+\frac{x^2}{2!}+\frac{x^3}{3!}+\frac{x^4}{4!}+\ldots,\quad x\in[0,\,1]$.

12. F(x)=$1-x+\frac{x^2}{2!}-\frac{x^3}{3!}+\frac{x^4}{4!}-\ldots,\quad x\in[2,\,3]$.

13. F(x)=$1+x^2+\frac{x^4}{2!}+\frac{x^6}{3!}+\frac{x^8}{4!}+\ldots,\quad x\in[-1,\,0]$.

14. F(x)=$1-\frac{x^2}{2!}+\frac{x^4}{4!}-\frac{x^6}{6!}+\frac{x^8}{8!}-\ldots,\quad x\in[1,\,2]$.

15. F(x)=$1-\frac{x^2}{3!}+\frac{x^4}{5!}-\frac{x^6}{7!}+\frac{x^8}{9!}-\ldots,\quad x\in[0,\,1]$.

16. F(x)=$-\dfrac{x}{2!}+\dfrac{x^3}{4!}-\dfrac{x^5}{6!}+\dfrac{x^7}{8!}-\ldots,\quad x\in[-1, 0]$.

17. F(x) = $2\left(\dfrac{x-1}{x+1}+\dfrac{(x-1)^3}{3(x+1)^3}+\dfrac{(x-1)^5}{5(x+1)^5}+\ldots\right),\quad x\in[1; 2]$.

18. F(x) = $\dfrac{x-1}{x}+\dfrac{(x-1)^2}{2x^2}+\dfrac{(x-1)^3}{3x^3}+\ldots,\quad x\in[1; 2]$.

19. F(x) = $(x-1)-\dfrac{(x-1)^2}{2}+\dfrac{(x-1)^3}{3}-\dfrac{(x-1)^4}{4}+\ldots,\quad x\in[0.5; 1.5]$.

20. F(x)=$\dfrac{\pi}{2}-\left(x+\dfrac{x^3}{2\cdot 3}+\dfrac{3x^5}{2\cdot 4\cdot 5}+\dfrac{3\cdot 5\cdot x^7}{2\cdot 4\cdot 6\cdot 7}+\dfrac{3\cdot 5\cdot 7x^9}{2\cdot 4\cdot 6\cdot 8\cdot 9}+\ldots\right),\quad x\in[-0.9; 0.9]$.

Задание 6

Для заданного натурального числа N:

1. найти количество всех делителей;
2. найти сумму всех делителей;
3. найти наибольший делитель, не совпадающий с самим числом N;
4. вывести на экран все делители, кратные целому числу C;
5. вывести на экран все делители, кратные целым числам C и D одновременно;
6. найти сумму всех делителей, не кратных целому числу C;
7. найти сумму всех делителей, кратных хотя бы одному из целых чисел C, или D;
8. найти среднее арифметическое всех делителей;
9. найти среднее арифметическое всех двузначных делителей;
10. найти среднее арифметическое всех делителей, попадающих в отрезок [a, b];
11. определить, является ли заданное число простым; если нет, то вывести на экран все его делители;
12. найти старшую цифру;
13. найти количество цифр;
14. найти среднее арифметическое значение цифр.

Задание 7

Даны два натуральных числа a и b:

1. сократить дробь вида a/b;
2. вычислить значение выражения $\dfrac{a}{b}+\dfrac{b}{a}$; результат представить в виде обыкновенной дроби, выполнив сокращение;
3. найти наибольшую цифру из старших цифр заданных чисел;
4. определить, в каком числе содержится больше значащих нулей;
5. определить, у какого числа больше делителей.

Задание 8

Вывести на экран все числа из отрезка [a; b]:

1. имеющие наибольшее количество делителей;
2. имеющие наименьшее количество делителей;
3. имеющие ровно K делителей;
4. сумма делителей которых кратна натуральному числу C;
5. сумма делителей которых является простым числом;
6. в записи которых встречается цифра C;
7. в записи которых ровно K четных цифр;
8. в записи которых все цифры различны.

Задание 9

Дано натуральное число N. Вывести на экран:

1. предшествующее по отношению к нему простое число;
2. ближайшее число, большее данного, сумма цифр которого кратна числу C;
3. ближайшее число, большее данного, сумма цифр которого кратна числам C и D одновременно;
4. ближайшее число, меньшее данного, сумма цифр которого кратна числу C;
5. ближайшее число, меньшее данного, сумма цифр которого кратна хотя бы одному из чисел C или D;
6. ближайшее число, большее данного, среди делителей которого есть число C;
7. ближайшее число, меньшее данного, среди делителей которого нет числа C;

Задание 10

На отрезке [a, b] найти все пары соседних натуральных чисел:

1. сумма которых является составным числом;
2. произведение которых является простым числом;
3. сумма которых образует симметричное число;
4. которые являются взаимопростыми числами;
5. наибольший общий делитель которых является простым числом.