4т-лекция

Операторы циклов языка C#

Вопросы:

1. Операторы присваивания

Оператор присваивания имеет две формы записи:

Полная форма:

имя_переменной = выражение;

Сначала вычисляется выражение, а затем результат присваивается имени переменной. Например:

```
y = (x + 2) / (3 * x) - 5;
```

С помощью одного оператора можно присвоить одно значение нескольким переменным, например: `x = y = z = 0;` `/* x, y, z=0 */`

или `z = (x = y) * 5;` Здесь сначала переменной **x** присваивается значение переменной **y**, далее вычисляется выражение **x*5**, и результат присваивается переменной **z**.

Сокращенная форма оператора присваивания:
имя_переменной операция= выражение;
где **операция** – одна из арифметических операций (+ , -, *, /, %); Например:

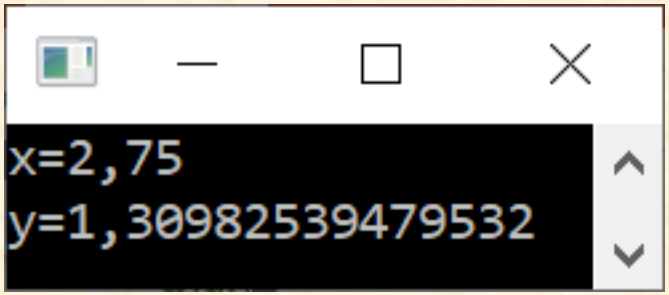
```
x*=5;        //  x=x*5;
s+=7;        // s=s+7;
y/=x+3;      // y=y/(x+3);
z%=3;        // z=z%3; определяет остаток
```

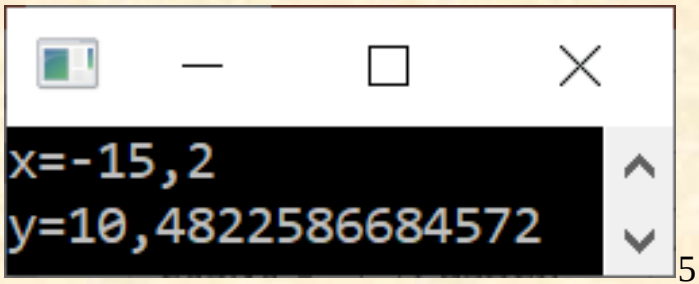
Сокращенная форма операции присваивания применяется тогда, когда переменная используется в обеих частях полной формы данного оператора.

Вычисление выражения (формулы)

- Вычислить значение выражения *y* при любых значениях аргумента: $y=(0,5x^2-1) / \sqrt{x * x + 1,5} + |10-e^{0,5x+1}| / \ln|2x+1,5|$

```
static void Main()
{   double x, y1, y2, y3, y;
    Console.Write("x=");
    x = double.Parse(Console.ReadLine());
    y1 = (0.5 * x * x - 1) / Math.Sqrt(x * x + 1.5);
    y2 = Math.Abs(10 - Math.Exp(0.5 * x + 1));
    y3 = Math.Log(Math.Abs(2 * x + 1.5));
    y = y1 + y2 / y3;
    Console.WriteLine("y=" + y);
}
```





2. Циклические операторы

В C# имеются 3 вида операторов цикла, их запись:

while (условие) {операторы} предусловие

do {операторы} while (условие); постусловие

for (инициализация; условие; модификация)
{ операторы }

foreach (тип имя-переменной in условное выражение)
{ операторы }

Имеются 4 типа операторов передачи управления:

- ✓ goto оператор безусловной передачи управления;
- ✓ break оператор выхода из цикла;
- ✓ continue оператор перехода к сл. шагу (итерации) цикла;
- ✓ return оператор возврата из функции.

6

Циклические операторы

Различают:

-итерационные циклы (while, do..while);

-арифметические циклы (for, foreach).

Группа действий, повторяющихся в цикле, называется его телом.

Однократное выполнение цикла называется его шагом.

В итерационных циклах известно условие выполнения цикла, количество повторений неизвестно.

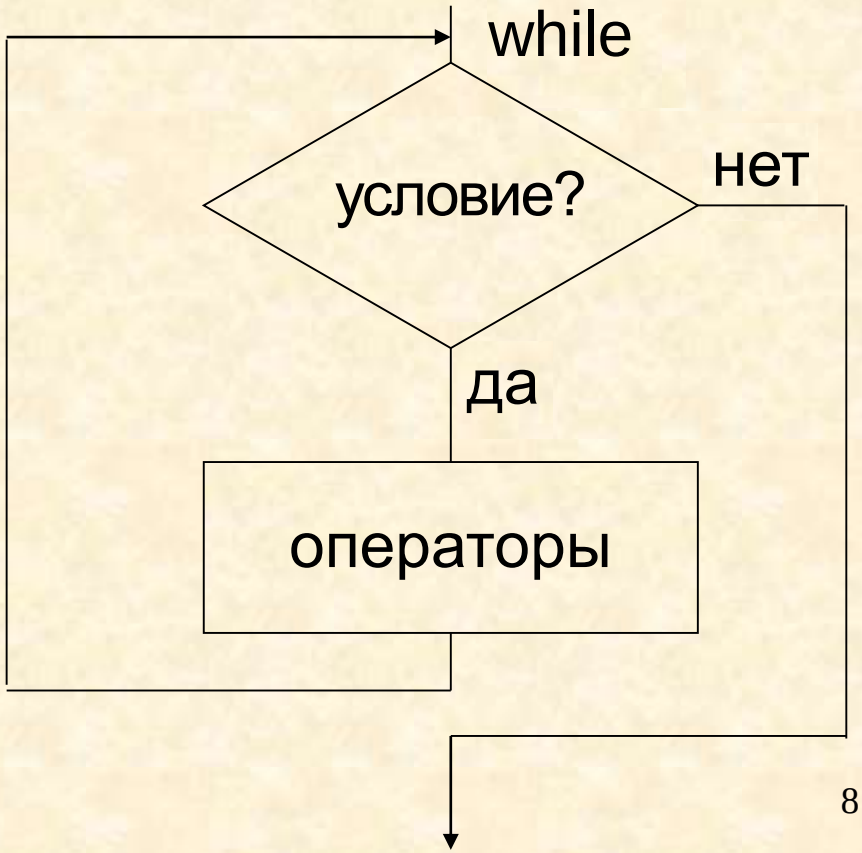
В арифметических циклах заранее известно количество повторений цикла. Она напоминает арифметическую прогрессию. Имеется параметр цикла, который изменяется от начального значения до конечного через определенный шаг.

3. Цикл с предусловием while

Оператор **while** организовывает цикл с предусловием. Он относится к итерационным циклам, где известно только условие выполнения цикла.

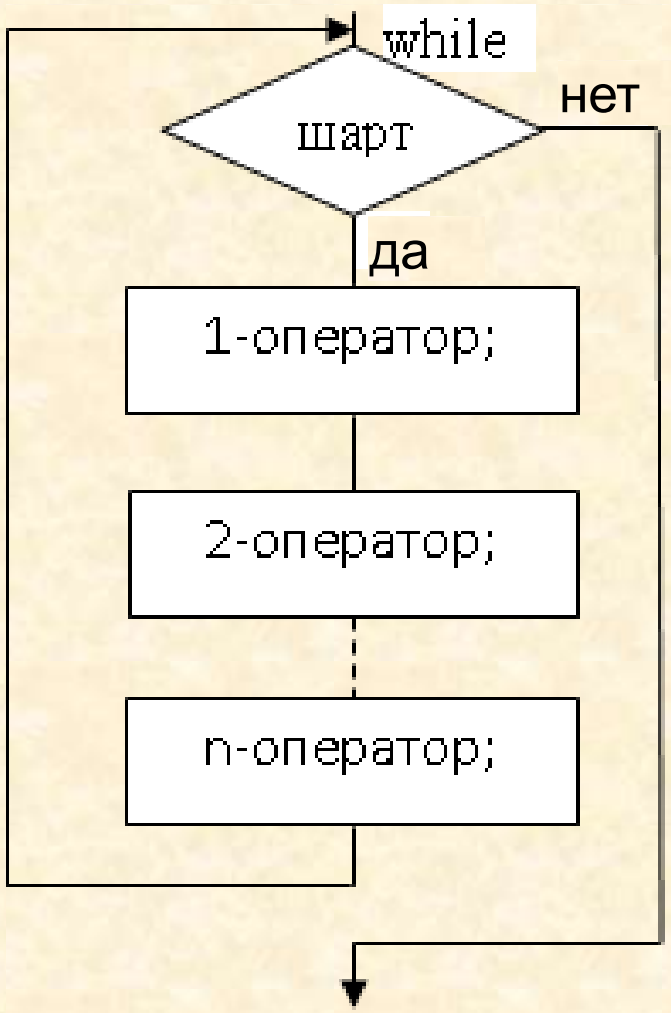
Формат записи:

```
while (условие)
{
   1 оператор;
   2 оператор;
   ......
   n оператор;
}
```



Когда в операторе повторяется блок операторов, тело цикла заключается в фигурные скобки:

```
while(выражение-условие)
{
 1 оператор;
 2 оператор;
 ......
 N оператор;
}
```



здесь выражение-условие или просто условие должно меняться внутри цикла.

Рассмотрим примеры.

Цикл с предусловием сначала проверяет условие, в качестве условия чаще всего используется отношение или логическое выражение. Если оно истинно, т. е. не равно 0, то тело цикла выполняется до тех пор, пока выражение не станет ложным.

Например:

```
while (a != 0)
{
    y = x + a;
    Console.WriteLine(y);  // вывод y на экран
    a =- 1;
}
```

Здесь переменная, которая входит в условие (*a*), должна изменяться внутри цикла.

Пример 1 (рис.1).

```
// определение суммы чисел от 1 до 100
static void Main()
{
   int s, k;
   s=0;  k=1;
   while (k<=100)
  {
     s+=k;
     k ++;
  }
  Console.WriteLine("s=" + s );
}
```

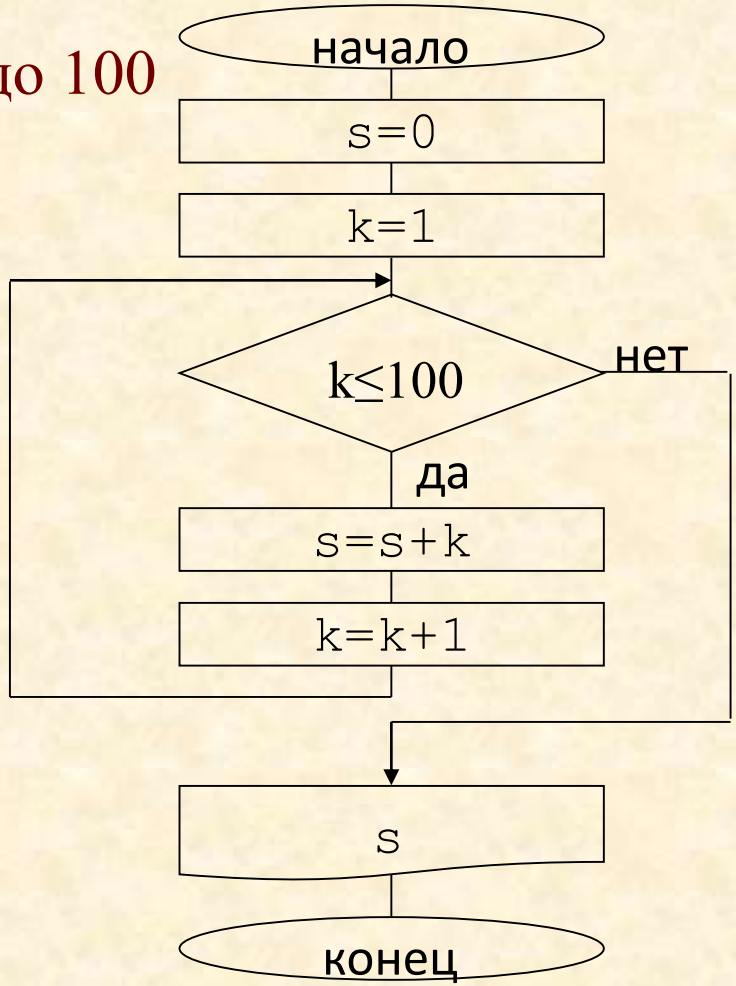
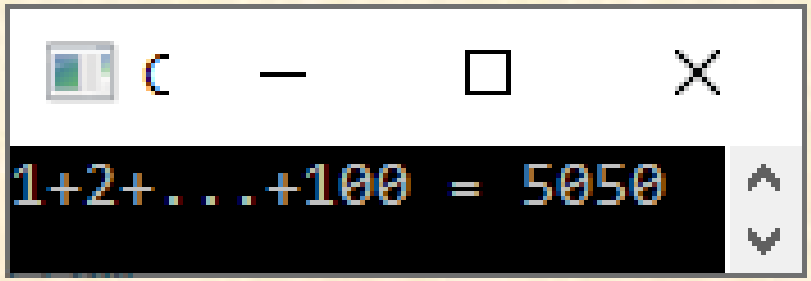
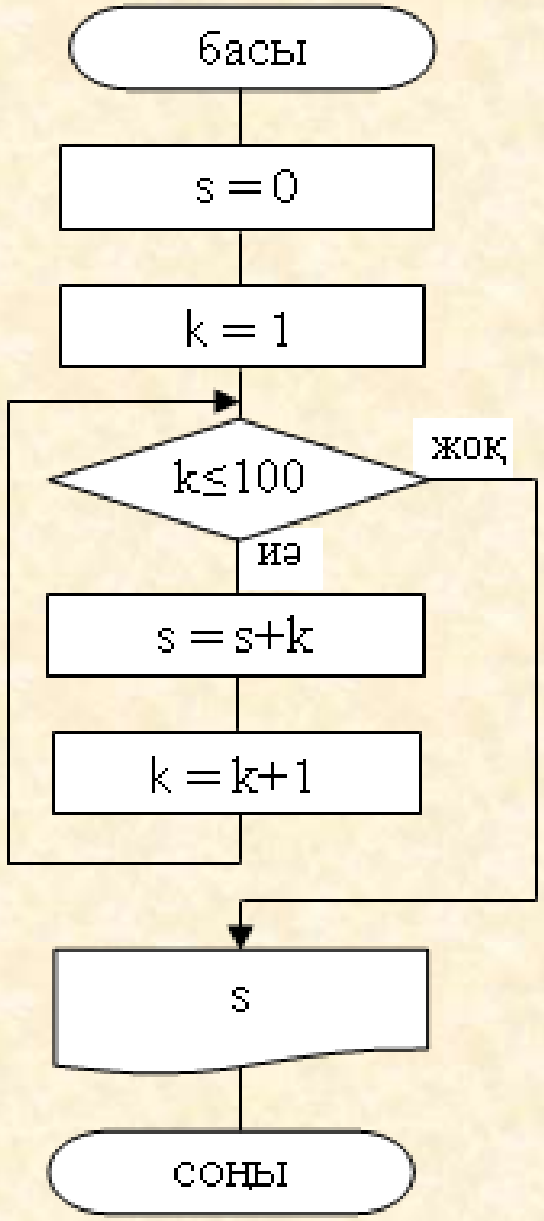


Рис. 1. Алгоритм сложения целых чисел от 1 до 100

Сумма целых чисел от 1 до 100

```
using System;
namespace Sum1
{  class Program
   {   static void Main()
       {   int s, k ;        //  s-сумма, k-переменная
                             //  управления циклом
           Console.Write("1+2+3+...+100 = ");
           s=0; k=1;         // значения переменных
           while (k <= 100)  // условие повторения
           {    s+=k;        //  s=0+1+2+...+100
                k+=1;        // немесе қысқаша  k++;
           }
           Console.WriteLine(s);
           Console.ReadKey();
       }
   }
}
```



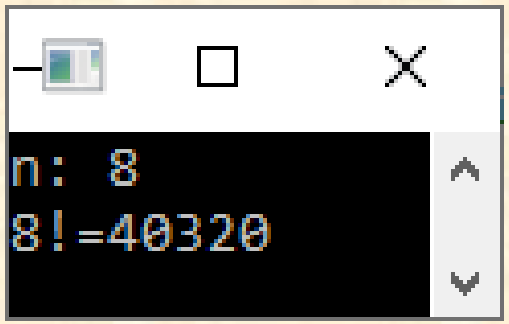


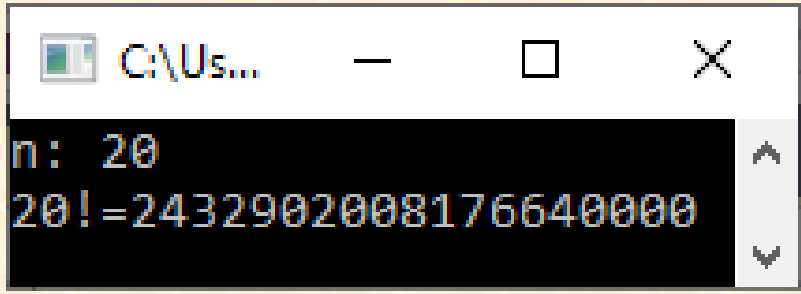
3.6-сурет. Бүтін сандарды қосу алгоритмі

Программа вычисления n!

n! = 1·2·3· … · n. Здесь сразу примем f = 1! =1. Поэтому n! вычисляется как произведение предыдущего значения f на следующее натуральное число.

```
using System;
namespace Factorial1
{  class Program
   {  static void Main()
      {  int  i, n;   long f=1;
         Console.Write("n: ");
         n = int.Parse(Console.ReadLine());
         f=1; i=2;
         while(i <= n)
         {  f=f*i;  i ++;  }
         Console.WriteLine($"{n}!={f} ");
         Console.ReadKey();
      }
   }
}
```



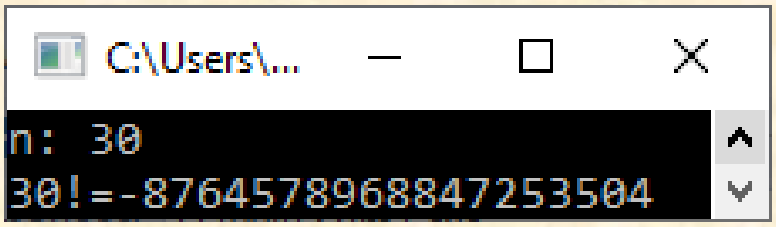


здесь знак **$** выводит значение переменной, указанной внутри фигурных скобок. Для значений n>25 программа выводит отрицательное значение, т.к. происходит переполнение разрядов.

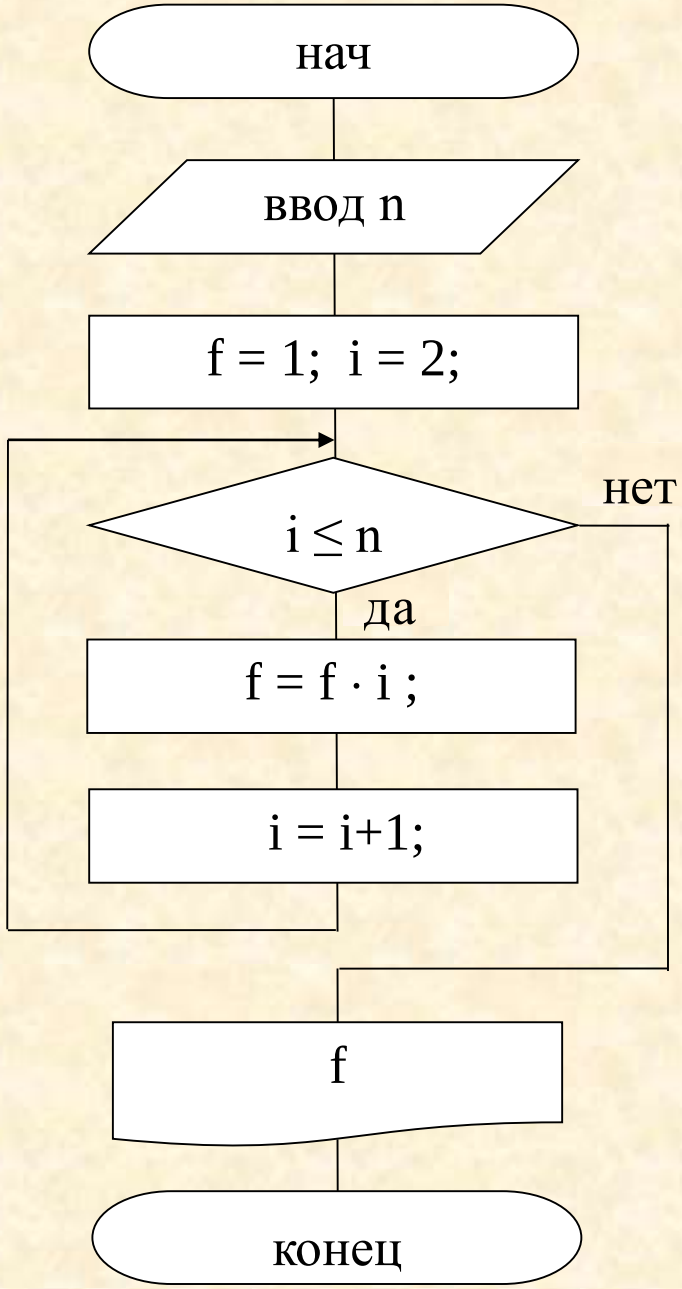
Блок-схема вычисления n!

$$n! = 1 \cdot 2 \cdot 3 \cdot \ldots \cdot n$$

```
using System;
class Program
  {  static void Main()
     {   int  i, n;   long f=1;
         Console.Write("n: ");
         n = int.Parse(Console.ReadLine());
         f=1;  i=2;
         while(i <= n)
         {  f=f*i; i ++;  }
         Console.WriteLine($"{n}!={f} ");
        Console.ReadKey();
     }
  }
```



Почему отрицательное число?



Алгоритм вычисления n!

```
// Определение количества разрядов целого числа
using System;
class WhileDemo
{  static void Main()
   {  int num = 435679, mag = 0;  // mag – кол-во разрядов
      Console.WriteLine("Число: " + num);
      while(num > 0)          // условие
      {  mag++;
         num = num/10;      // урезание разрядов числа
      };
      Console.WriteLine("Количество цифр: " + mag);
   }
}
```

Число делим на 10 6 раз, тогда получим 0, тогда цикл завершается, например:

435679/10=43567 43567/10=4356…435…43…4…0

1 2 3 4 5 6 раз

Результат: Количество цифр: 6

```
/* определение значений функции y = -2.4x^2+5x-3 x при измене-
ниях x от x0 до xk с шагом dx */
static void Main()
 {  float x, y, x0, xk, dx;
    Console.WriteLine("x0,xk,dx: ");
    x0 = float.Parse(Console.ReadLine());
    xk = float.Parse(Console.ReadLine());
    dx = float.Parse(Console.ReadLine());
    Console.WriteLine("----------------");
    Console.WriteLine("  x     |   y ");
    Console.WriteLine("----------------");
    x = x0;
    while (x <= xk)
     {  y = (float) -2.4 * x * x + 5 * x - 3;
        Console.WriteLine(" {0,4:f1} \t|{1,7:f2}",x,y);
        x = x + dx;
     }
     Console.WriteLine("----------------");
 }
```

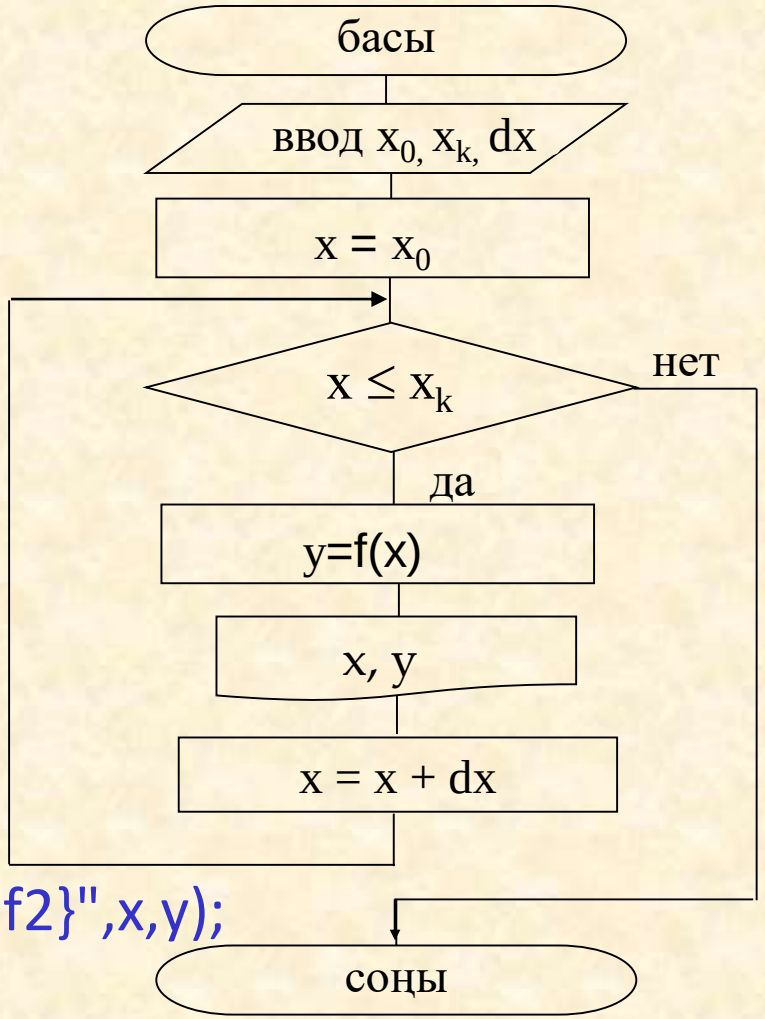


Схема табулирования функции

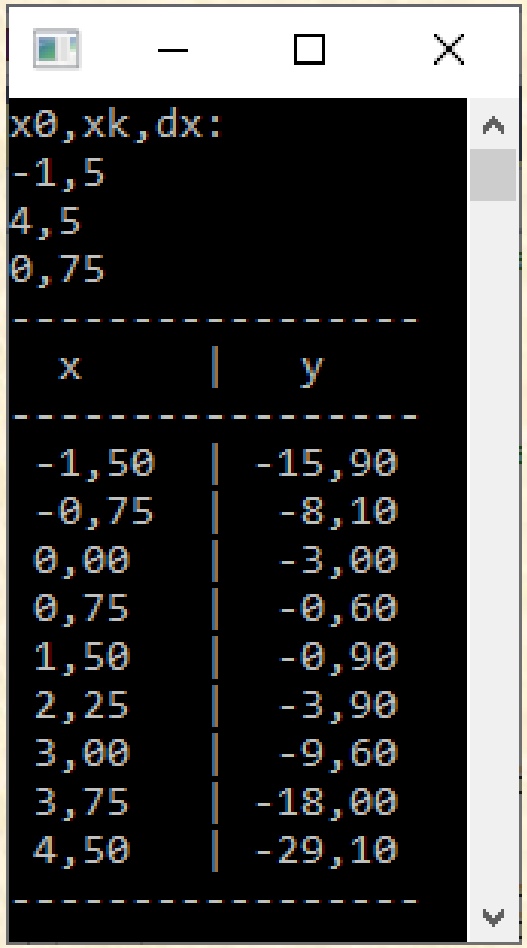
Результат программы:

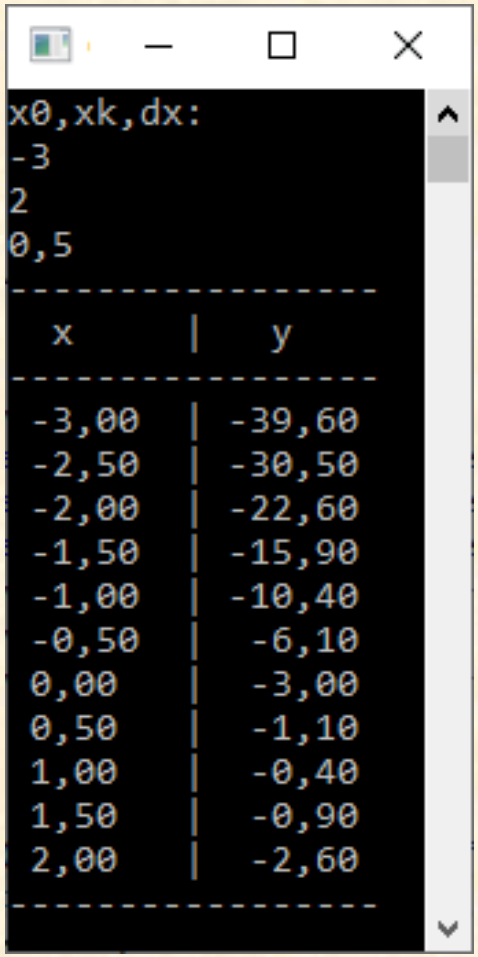
Количество повторов (2-лекция):

$$n = \left[\frac{xk - x0}{dx}\right] + 1$$

$$n = \frac{4,5 - (-1,5)}{0,75} + 1 = 9$$

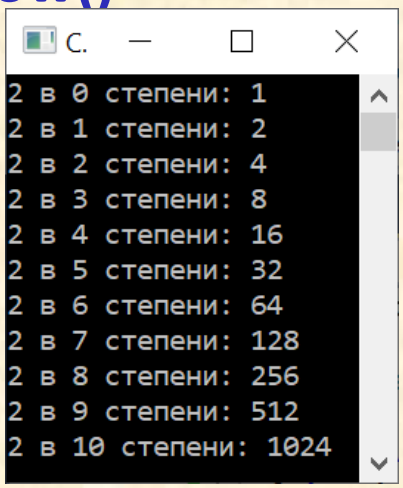
$$n = \frac{2 - (-3)}{0,5} + 1 = 11$$





В цикле while условие проверяется в начале цикла, поэтому он может не выполняться ни разу, если условие ложное. В сл. примере вычисляются степени числа 2 от 0 до 10 , и если условие изменяется на дополнительное условие (i < 0), цикл не выполняется ни разу.

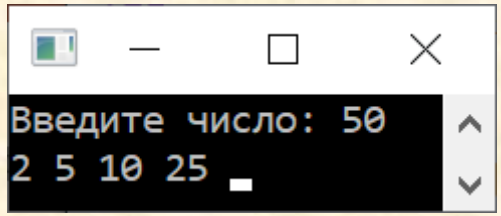
```
// Вычисление степени 2 без функции Mat.Pow()
using System;
class Power1
 {  static void Main()
      {  int i = 0, result=1; // result - результат
         while (i <= 10)      // доп. условие i < 0
         {
            Console.WriteLine("2 в " + i + " степени: " + result);
            result *= 2;   i++;
         }
      }
}
```

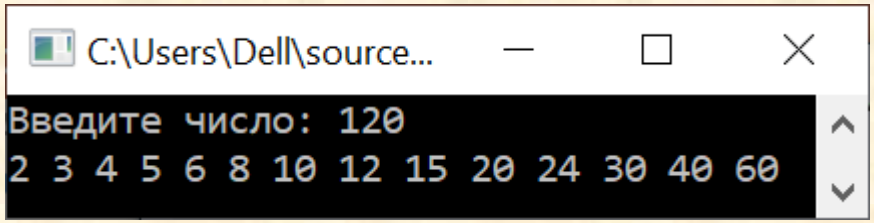


Здесь при условии i<=10, выполняется цикл while, а при условии i<0, цикл while не выполняется ни разу.

В сл. примере определяются все делители введенного целого числа num. Максимальный делитель - это половина заданного числа.

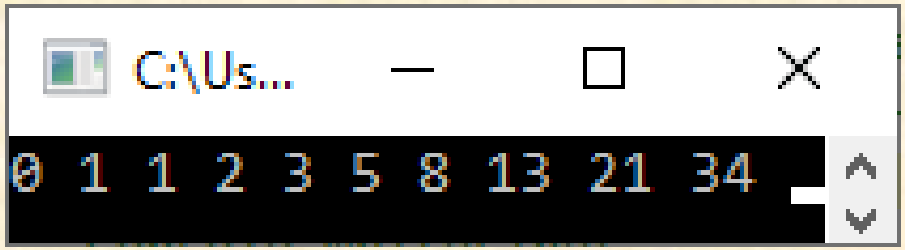
```
// Определение делителей положительного целого числа
  ...
 static void Main()
  {  int num;
     Console.Write("Введите число: ");
     num = int.Parse(Console.ReadLine());
     int half = num / 2;  // половина числа
     int div = 2;              // наименьший делитель
     while (div <= half)
      {  if ((num % div) == 0) Console.Write(div+" ");
         div++;
      }
  }
```





Опредение чисел Фибоначчи (f_i <50) $f_{i+1} = f_i + f_{i-1}$

```
using System;
namespace Fibonachi50        // 0,  1,  1,  2,  3,  5, …
{ class Program
  {  public static void Main()
    {  int  oldNumber = 0;      // первые два числа: 0 и 1
       int  currentNumber = 1;
       int  nextNumber;
       Console.Write(currentNumber + " ");
       while (currentNumber < 50)
       {  Console.Write(currentNumber + " ");
          nextNumber = currentNumber + oldNumber;
          oldNumber = currentNumber;
          currentNumber = nextNumber;
       }
    }
 }
```



20

4. Цикл с постусловием do .. while

Оператор цикла do .. while называется оператором цикла с постусловием и используется в тех случаях, когда необходимо выполнить тело цикла хотя бы один раз.

Этот оператор как и оператор while используется тогда, когда заранее не известно количество повторений тела цикла. Формат оператора имеет следующий вид:

```
do
    оператор;
while (выражение);
```

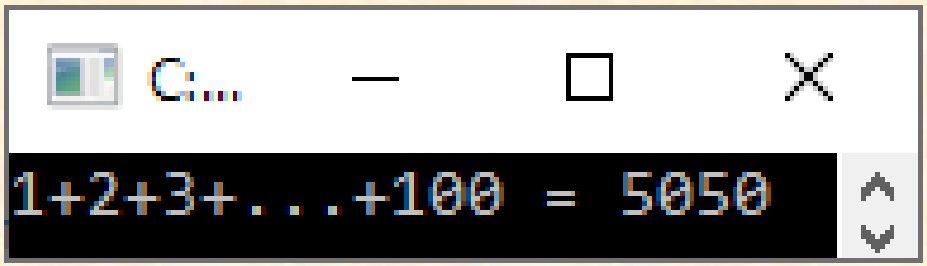
В качестве оператора может стоять составной оператор (блок).

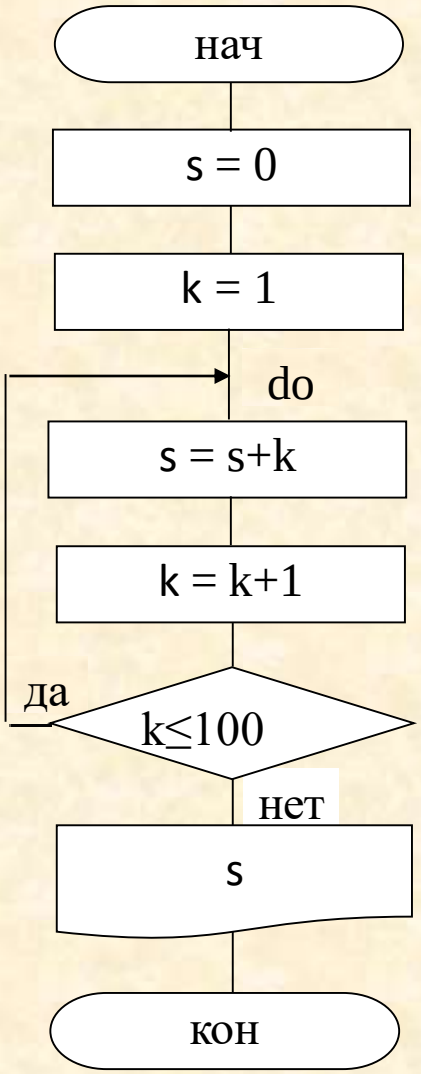
Для составного оператора блок-схема и формат записи:

```
do
 {
        1 оператор;
        2 оператор;
        ......
        n оператор;
  }
 while (условие);
```

Тело цикла выполняется до тех пор, пока условие истинно. Тело цикла выполняется хотя бы раз, т.к. проверка условия в конце цикла.

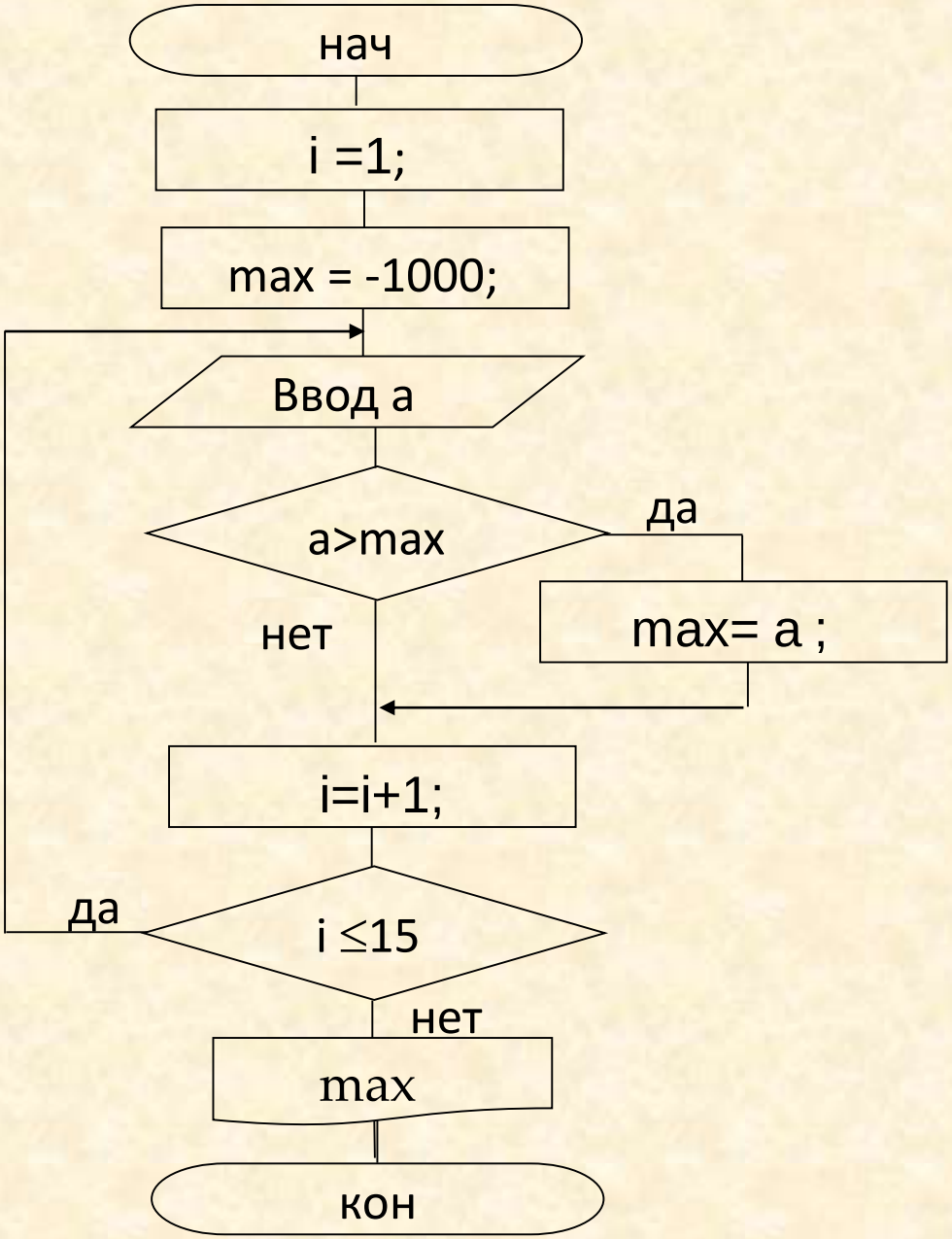
```
// Определение суммы чисел от 1 до 100
using System;
class Program
 {   static void Main()
     {  int s, k ;   //  s-сумма, k-переменная
                     // управления циклом
        Console.Write("1+2+3+...+100 = ");
        s=0; k=1;  // значения переменных
        do          // начало цикла
        {   s+=k;   //  s=0+1+2+...+100
            k++;    // увеличение k на 1;
        }
        while (k <= 100);   // условие повтора
        Console.WriteLine(s);
        Console.ReadKey();
     }
 }
```



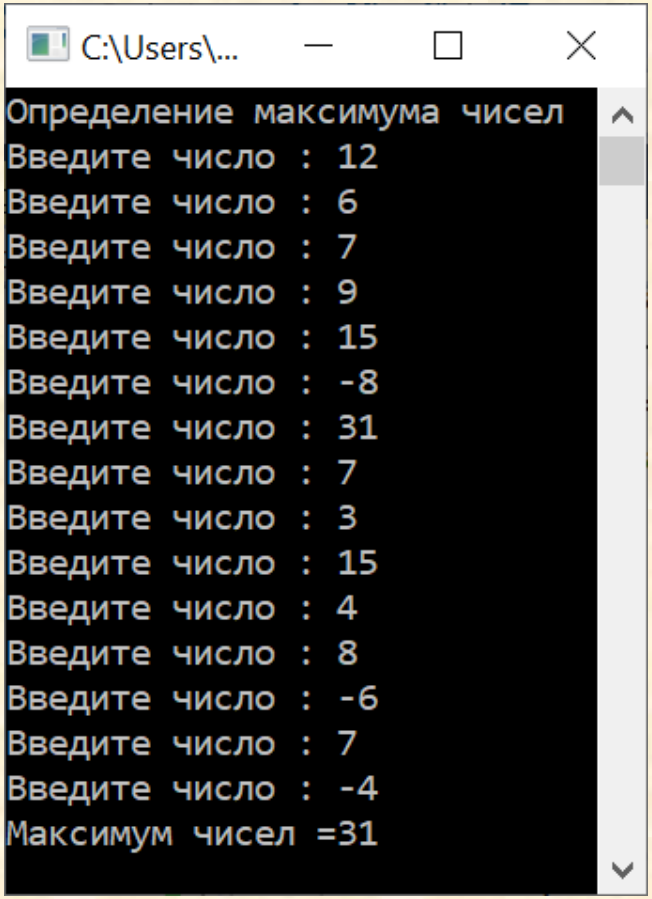


Бүтін сандарды қосу алгоритмі

Пример 4. Определение максимума из 15 чисел

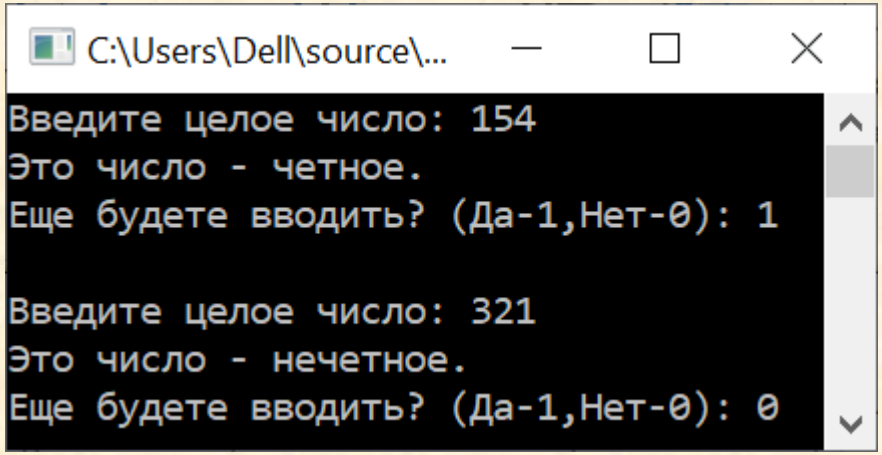


```
// Программа определения максимума из 15 чисел
static void Main()
   {   int a, i, max;
       Console.WriteLine("Определение максимума чисел");
       i = 1; max = -1000;
       // присваиваем максимуму минимальное целое
       do
       {
          Console.Write("Введите число : ");
          a = int.Parse(Console.ReadLine());
          if (a > max) max = a;
          i++;
       }
       while (i <= 15);
       Console.WriteLine("Максимум чисел ="
                                  + max);
}
```



Пример 5. В следующей программе определяется четность (нечетность) введенного целого числа.

```
static void Main ()
{
   int k;      // вводимое целое число
   int s;      // признак повтора
   do
    { Console.Write("Введите целое число: ");
      k=int.Parse(Console.ReadLine());
      Console.Write("Это число – ");
      if (k % 2 == 0)
         Console.WriteLine( "четное.");
      else
         Console.WriteLine( "нечетное." << endl;
      Console.Write( "\nЕще будете вводить? Да - 1, Нет - 0: ");
      symbol = Console.ReadLine();
    }
    while (s == 1);
}
```



Пример 6. Определить следующую бесконечную сумму

$$s = \sum_{i=1}^{\infty} \frac{1}{i^2} = 1 + \frac{1}{2^2} + \frac{1}{3^2} + \frac{1}{4^2} + \frac{1}{5^2} + \ldots + \frac{1}{n^2} + \ldots$$

с заданной точностью $\varepsilon = 10^{-5}$, т.е. нужно завершить суммирование, когда очередной член ряда станет меньше чем точность – ε.

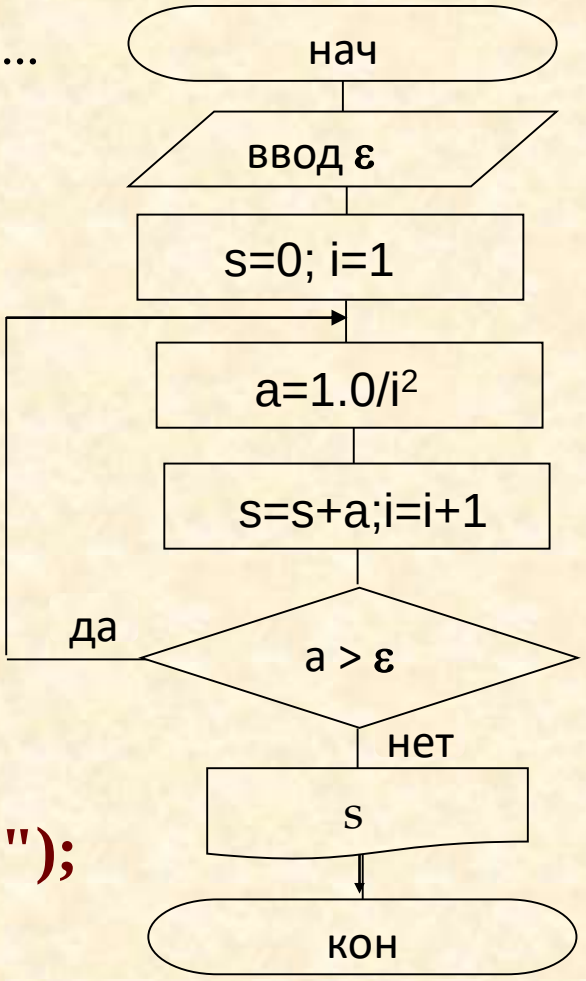
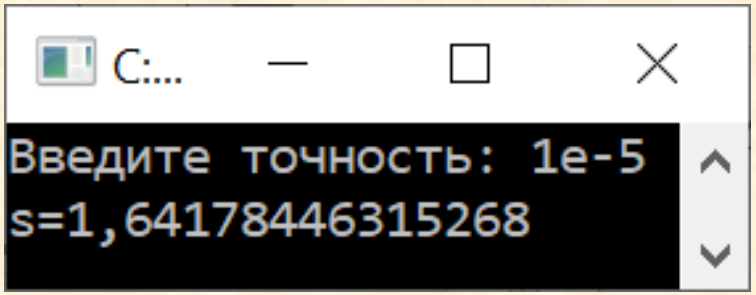


Рис.4. Схема вычисления бесконечной суммы

```
using System;
class Program
{  static void Main ( )
   {   int i, float s, eps;
       Console.Write("Введите точность: ");
       eps=int.Parse(Console.ReadLine());
       s=0; i=1;
```

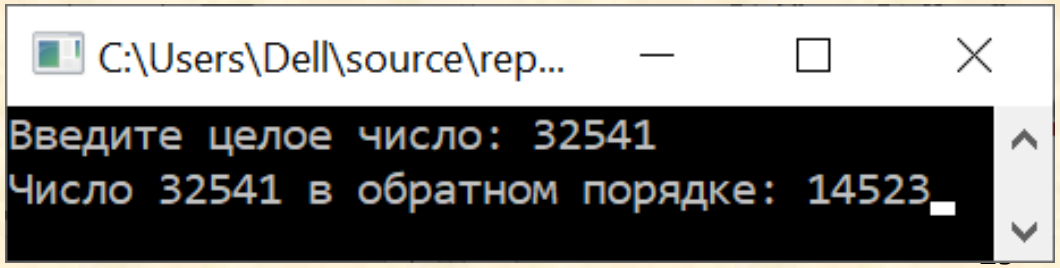
```
double a;
do
   {
      a=1.0/i/i;     // или a=1.0/(i*i)
      s+=a;
      i++;
   }
 while (a>eps);
 Console.Write("s=" + s);
}
```

Результат работы программы:



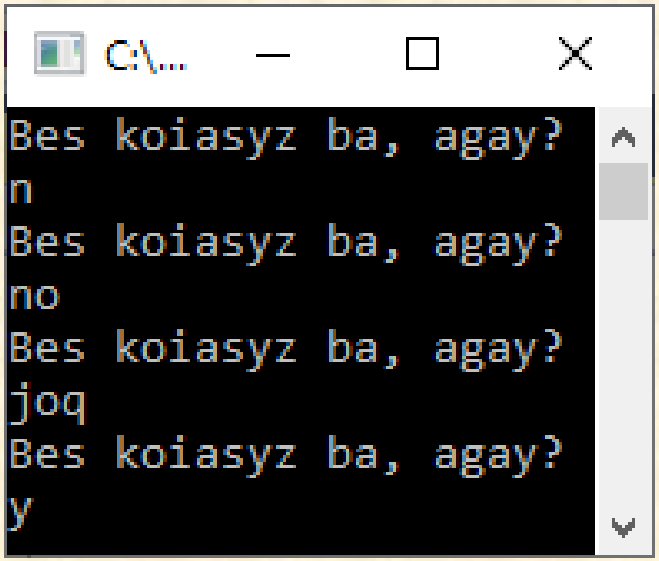
Запись целого числа в обратном порядке

```
// Цифры целого числа записываются в обратном порядке
using System;
class DoWhileDemo
{   static void Main()
    {  int num, nextdigit;
       Console.Write("Введите целое число: ");
       num = int.Parse(Console.ReadLine());
      Console.Write("Число " + num + " в обратном порядке: ");
       do
       {   nextdigit = num % 10;        // последняя цифра
           Console.Write(nextdigit);   // и ее вывод на экран
           num = num / 10;             // урезание последней цифры
       }  while (num > 0);

       Console.ReadKey();
    }
}
```



do … while циклі (цикл с постусловием)

```
// Повторить вопрос до утвердительного ответа y (yes)
using System;
namespace Baga5
{  class Program
   {  static void Main()
      {  char answer;
         do
         {   Console.WriteLine("Bes koiasyz ba, agay?");
             answer = (char)Console.Read();
             Console.ReadLine();
         } while (answer != 'y');
         Console.ReadKey();
      }
   }
}
```



5. Цикл с параметром

Этот оператор цикла применяется тогда, когда заданы начальное, конечное значения параметра цикла, а также его шаг изменения, при этом один или несколько действий повторяется несколко раз. Форма записи оператора:

for (выражение_1; выражение-условие_2; выражение_3)
{ <операторы> };

Выражение_1 – задает начальные условия для цикла (инициализация). Выражение-условие_2 определяет условие выполнения цикла, если оно не равно 0, цикл выполняется, а затем вычисляется значение выражения_3. Выражение_3 – задает изменение параметра цикла.

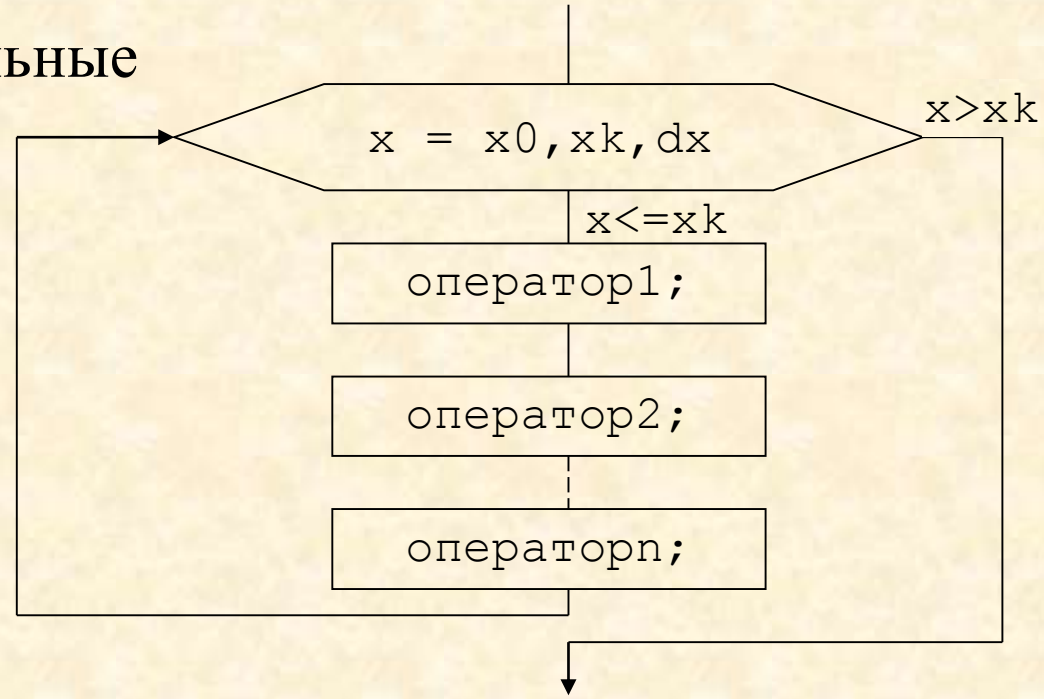


Рис.5. Алгоритм выполнения цикла for

Выражение_1 и выражение_3 могут состоять из нескольких выражений, разделенных запятыми. Цикл продолжается до тех пор, пока выражение-условие не станет равно 0. Любое выражение может отсутствовать, но разделяющие их « ; » должны быть обязательно.

Для параметра цикла *x* с начальным и конечным значениями *x*0, *xk* и шагом изменения *dx* (рис. 3.1) оператор for запишется так:

```
for (x=x0; x<=xk; x=x+dx)
   { <оператор1>;
     <оператор2>;
       . . .
     <операторn>;
   }
```

Другие примеры использования цикла с параметром.

1) Уменьшение параметра:

```
for (n=10; n>0; n--)
{ <операторы> };
```

2) Изменение шага корректировки:

```
for (n=2; n>60; n+=13)
{ <операторы> };
```

3) Возможность проверять условие отличное от условия, которое налагается на число повторений:

```
for (num=1;num*num*num<216; num++)
{ <операторы> };
```

Пример7. Определение суммы целых чисел до 100

```
using System;
 class Program
 {  static void Main ( )
    {   int s=0, i;
        for (i=1; i <= 100; i ++)
        s += i;
        Console.WriteLine("s=" + s);
        Console.ReadLine( );
    }
 }
```

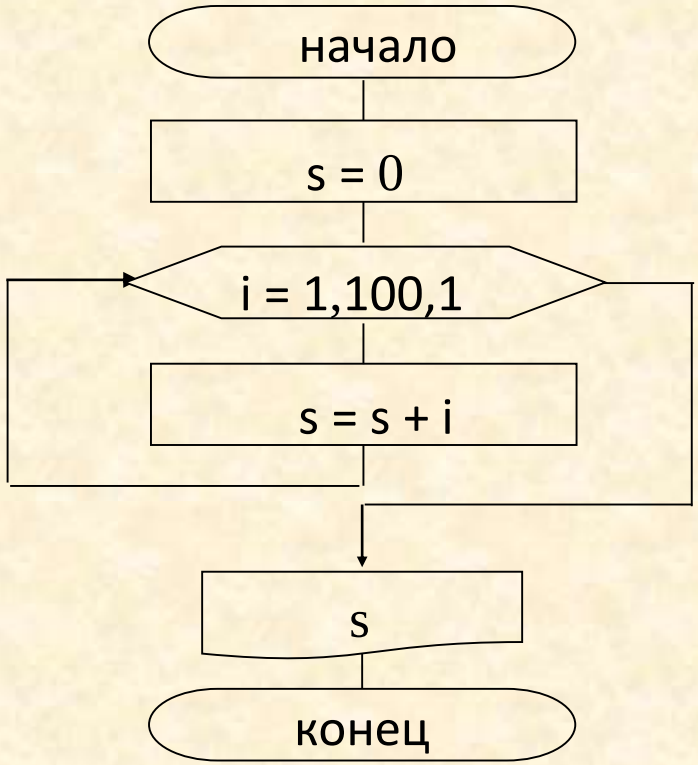
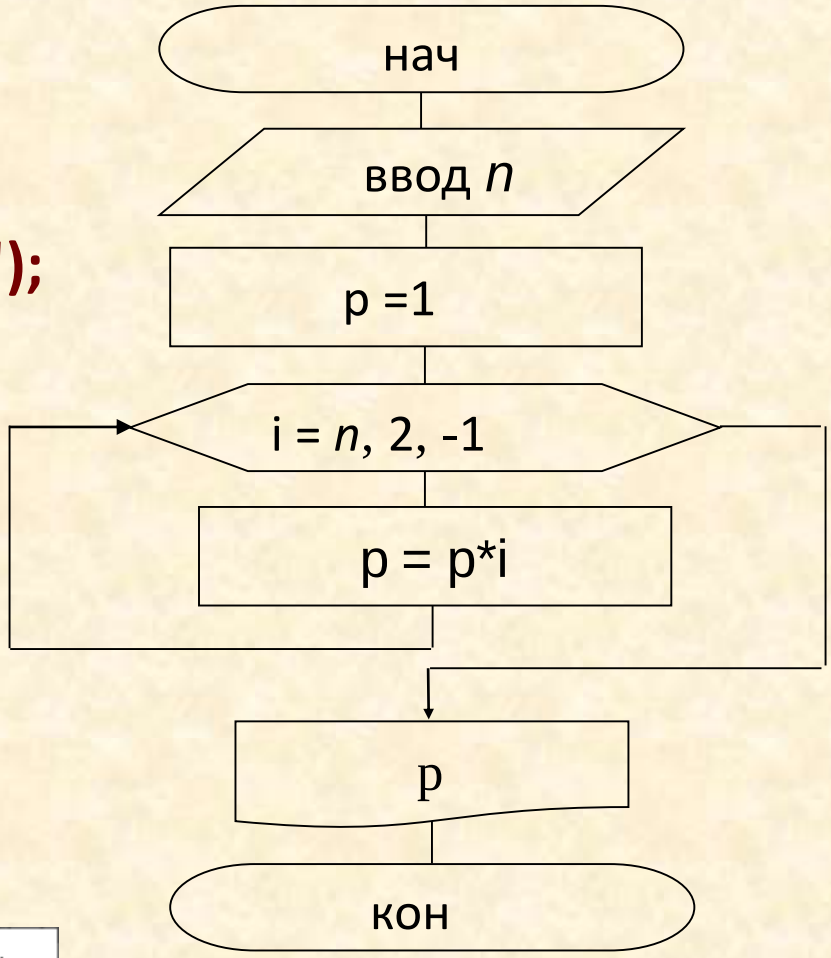


Рис.6. Алгоритм определения суммы

Пример 8. Определить произведение натуральных чисел – *n*!, где *n*!=1*2*... *n*. Здесь используем обратный ход изменения параметра цикла (рис.7).

```
static void Main()
 {
    int p = 1, i, n;
    Console.Write("Введите число n: ");
    n = int.Parse(Console.ReadLine());
    Console.Write("Произведение
                        чисел от 1 до n: ");
    for (i = n; i > 1; i-- )
        p *= i;
    Console.WriteLine(" s = " + p);
    Console.ReadLine();
 }
```



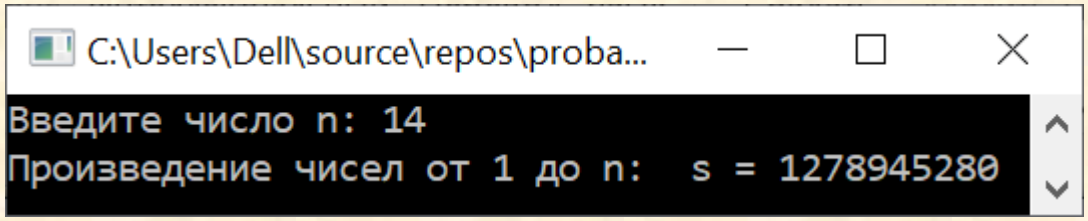


Рис.7. Алгоритм определения факториала чисел

6. Операторы перехода

Операторы перехода выполняют безусловную передачу управления.

break - оператор прерывания цикла по условию.

```
...
   <операторы>
  if (<выражение_условие>)  break;
  <операторы>
...
```

т. е. оператор break целесообразно использовать, когда условие продолжения итераций надо проверять в середине цикла.

Пример:

```
/* ищем сумму чисел вводимых с клавиатуры до
тех пор,  пока не будет введено 100 чисел или 0 */
int  i, x;
for(s=0, i=1; i<100; i ++)
{
    Console.WriteLine("введите  x: ");
    x = int.Parse(Console.ReadLine());
    if (x==0)  break;   // если ввели 0, то
                        // суммирование заканчивается
     s+=x;
}
```

6.1. Оператор continue

continue – переход к следующей итерации цикла. Он используется, когда тело цикла содержит ветвления.

Пример:

```
/*   ищем количество и сумму положительных
     чисел  */
for( k=0,s=0,x=1; x!=0; )
{   Console.WriteLine("введите  x: ");
    x = int.Parse(Console.ReadLine());

    if  (x<=0)   continue;
    k++;
    s+=x;
}
```

6.2. Оператор goto

Оператор goto имеет формат:

goto метка;

В теле той же функции должна присутствовать конструкция:

метка: оператор;

Метка – это обычный идентификатор, областью видимости которого является функция. Оператор goto передает управления оператору, стоящему после метки.

Использование оператора goto оправдано, если необходимо выполнить переход из нескольких вложенных циклов или переключателей вниз по тексту программы или перейти в одно место функции после выполнения различных действий.

6.3. Оператор return

Оператор return – оператор возврата из функции. Он всегда завершает выполнение функции и передает управление в точку ее вызова.

Вид оператора:

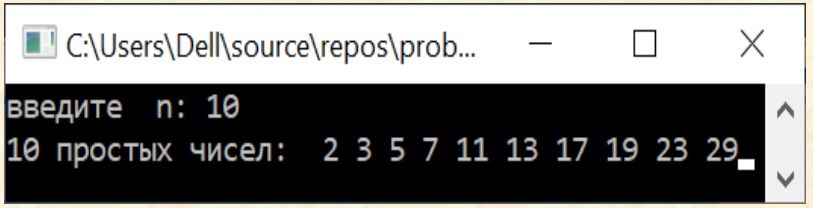
return [выражение];

7. Вложенные циклы

Часто в один цикл входит другой цикл, внутренний цикл, в свою очередь, может содержать еще цикл и т.д. Таким образом, будут организованы вложенные циклы.

Пример 10: Напечатать N простых чисел.

```
 static void Main( )
{  int a = 1, n, d;
   Console.Write("введите  n: ");
   n = int.Parse(Console.ReadLine());
   Console.Write(n +" простых чисел: ");
   for (int i = 0; i < n;)      // внешний цикл
    {  a++; d = 1;
       do                       // внутренний цикл
          { d++; }
       while (a % d != 0);     //конец внутреннего цикла
       if (a == d)   {  Console.Write(" " + a);   i++;  }
    }                           // конец внешнего цикла
}
```

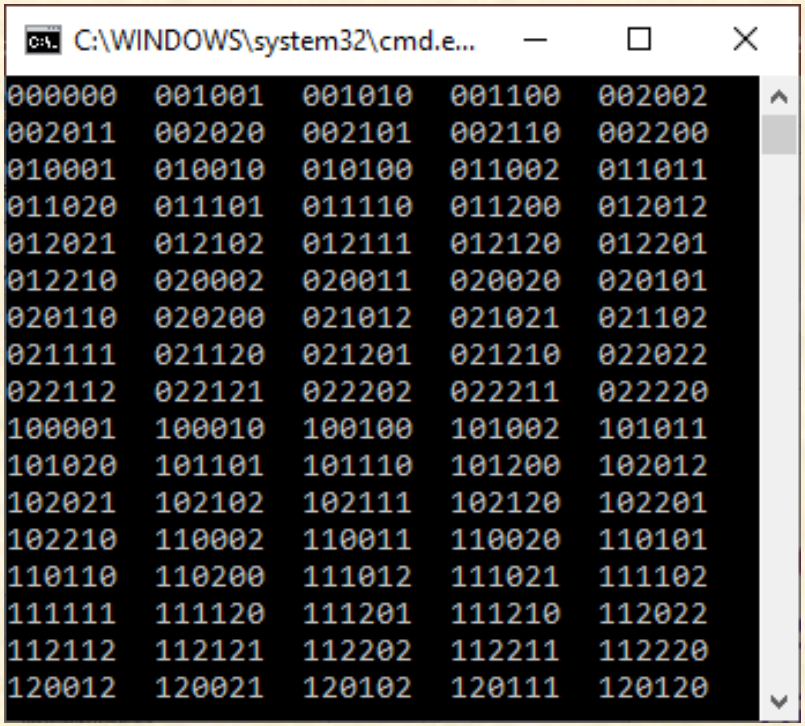


/* Определить номера счастливых билетов, в которых сумма первых трех разрядов совпадает суммой второй тройки разрядов */

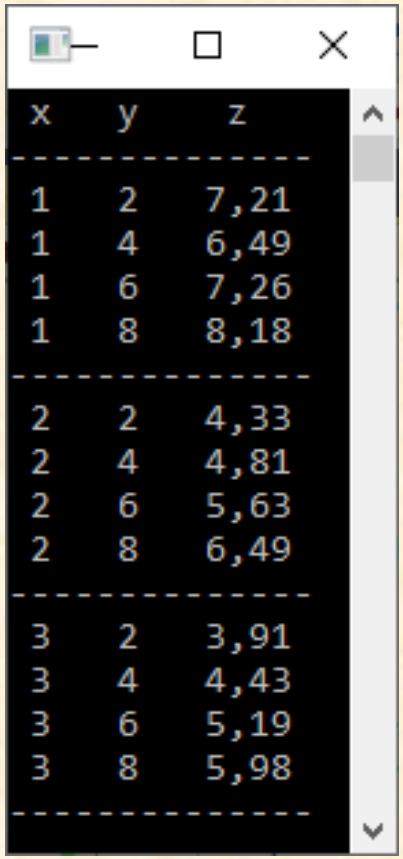
```
…
static void Main( )
{ int a,b,c,x,y,z;

   for (a=0; a<9; a++)
   for (b=0; b<9; b++)
    for (c=0; c<9; c++)
     for (x=0; x<9; x++)
      for (y=0; y<9; y++)
       for (z=0; z<9; z++)
            if (a+b+c==x+y+z)
                Console.Write($"{a}{b}{c}{x}{y}{z}   ");
}
```

/* Определить номера счастливых билетов, в которых сумма первых трех разрядов равна сумме второй тройки разрядов */

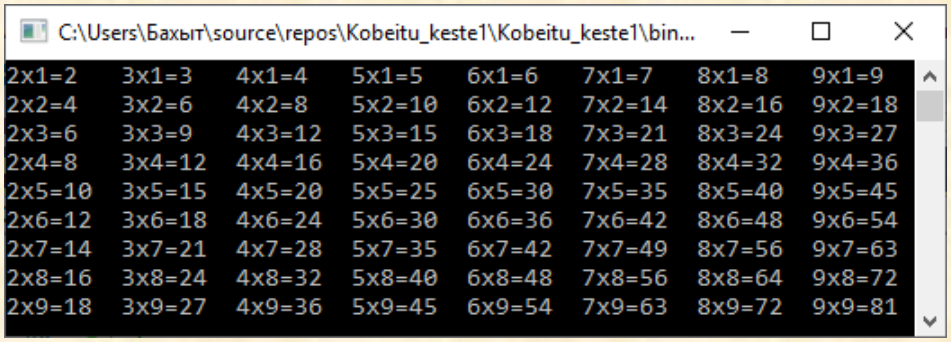
```
using System;
using System.Threading;
namespace Happy_bilet1
{ class Program
   {  static void Main()
      {  int a,b,c,x,y,z;
         for (a=0; a<=9; a++)
            for (b=0; b<=9; b++)
               for (c=0; c<=9; c++)
                  for (x=0; x<=9; x++)
                     for (y=0; y<=9; y++)
                        for (z=0; z<=9; z++)
                           if (a+b+c==x+y+z)
                              {  Console.Write($"{a}{b}{c}{x}{y}{z}  ");
                                 Thread.Sleep(100);  // задержка
                              }
      }
   }
}
```



```
// При x=1,2,3 и y=2,4,6,8, вычислить функцию z=(2y+x)/ln(xy)
 using System;
 class Program
 {   static void Main()
     {  int x, y;
        double z;
        Console.WriteLine(" x    y    z");
        Console.WriteLine("---------------");
        for (x = 1; x <= 3; x++)  // начало внешнего цикла
        {    y = 2;
             while (y <= 8)        // начало внутреннего цикла
             {   z = (2 * y + x) / Math.Log(x * y);
                 Console.WriteLine(" {0}  {1,2}{2,6:f2}", x, y, z);
                 y += 2;                // конец внутреннего цикла
             }
             Console.WriteLine("---------------");
        }                               // конец внешнего цикла
        Console.ReadKey();
     }
}
```



```
//  Таблица умножения для чисел от 2 до 9
using System;
namespace Table1
{ class Program
   {  static void Main()
      {   int m;      // число, указывающее таблицу умножения
          int n;      // множитель

          for (n = 1; n <= 9; n++)          // начало внешнего цикла
          {   for (m = 2; m <= 9; m++)  // начало внутреннего цикла
              {
                Console.Write($"{m}x{n}={m * n}\t");
              }                                    // конец внутреннего цикла
              Console.WriteLine();
          }                                        // конец внешнего цикла
          Console.ReadKey();
      }
   }
}
```



```
//  Цветная таблица умножения для чисел от 2 до 9
using System;
namespace Table1
{  class Program
   {  static void Main()
      {   int m;      // число, указывающее таблицу умножения
          int n;      // множитель
          Console.BackgroundColor = ConsoleColor.Yellow;
                                                 // желтый цвет фона
          Console.ForegroundColor = ConsoleColor.Red;
                                                // красный цвет шрифта
          for (n = 1; n <= 9; n++)       // начало внешнего цикла
          {  for (m = 2; m <= 9; m++) // начало внутреннего цикла
             {  Console.Write($"{m}x{n}={m * n}\t");
             }                                  // конец внутреннего цикла
             Console.WriteLine();
          }                                    // конец внешнего цикла
          Console.ReadKey();
      }
   }
}
```

Результат программы

Спасибо за внимание!