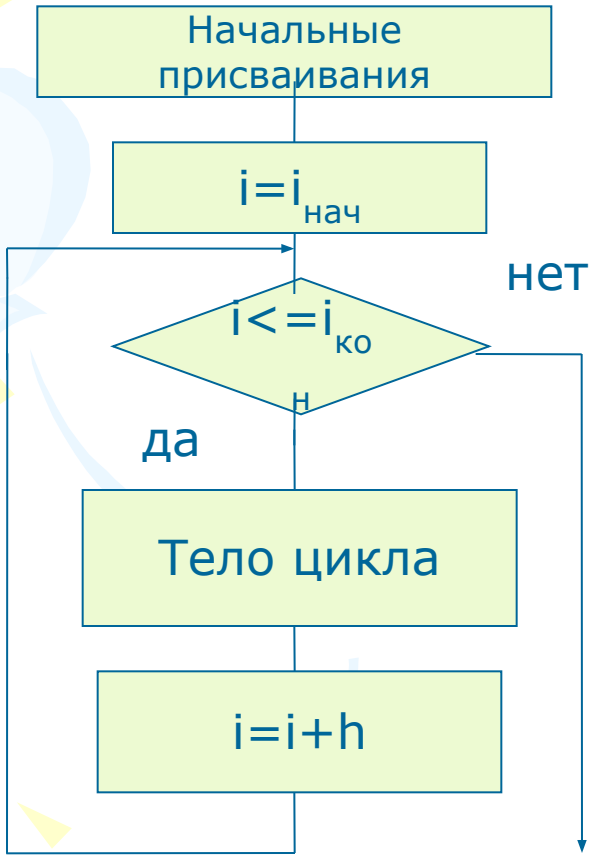
# Лекция 2
# Тема: Организация циклических программ. Циклы по счетчику. Циклы по условию. Вложенные циклы

1. Циклы по счетчику. Использование рекуррентных соотношений для вычисления члена суммы
2. Циклы по условию. Средства языка C# для организации циклов по условию
3. Вычисление рядов. Вложенные циклы

# Циклы по счетчику

- количество повторений цикла **определено до** начала выполнения цикла




- При организации **цикла по счетчику** необходимо:
- выделить повторяющиеся действия и записать их в общем виде (тело цикла);
- выбрать **управляющую переменную цикла**. Это может быть какая-либо величина, имеющаяся в постановке задачи, либо используемая специально в качестве счетчика
- определить **параметры цикла**, т.е. начальное и конечное значения управляющей переменной и шаг ее изменения
- Использовать оператор `for`.

**Пример. Вычислить сумму $s = 3 + 3^2 + 3^3 + \ldots + 3^8$.**

На каждом шаге алгоритма необходимо прибавлять к сумме очередное слагаемое ($s = s + 3^i$), где $i = 1, 2, 3, \ldots 8$.

Обратим внимание на то, что следующий член суммы может быть получен из предыдущего домножением его на 3 (возведение в степень в языке отсутствует). Для реализации этой возможности необходимо запоминать значение очередного слагаемого в какой-либо переменной (например, a):

```
int s = 0, a = 1;
for (int i = 1; i <= 8; i = i + 1)
{
   a = a * 3;
   s = s + a;
}
Console.WriteLine("s = {0}", s);
```

Выражение для получения очередного члена последовательности из предыдущего называется ***рекуррентной формулой***.

Flash05.sw

# Пример

Вычислить $$s = \frac{2}{3}\sum_{i=1}^{25}(-1)^i \frac{2i}{i^2+2}$$

- Вычисление каждого члена суммы целесообразно осуществлять в два приема. Сомножитель $(-1)^i$ (обозначим его через p) может вычисляться **рекуррентно**. Для следующего члена необходимо вычислять p = -p и далее очередной член суммы получать умножением его абсолютной величины на p. Программа, таким образом, будет иметь вид

```
double s = 0.0, a;
int i, p = 1;
for (i = 1; i <= 25; i = i + 1)
{
   p = -p;
   a = p * 2 * i / (i * i + 2.0);
   s = s + a;
}
s = (2.0 / 3.0) * s;
Console.WriteLine("{0:f4}",s);
```


Здесь использован вывод по формату: значение s выводится с 4 знаками в дробной части.

Flash05.swf

# Пример

Вычислить s = 5/8 + 7/10 + ... + 31/34.

Для организации цикла в данном случае можно использовать специальную переменную (счетчик), которая определяет количество проходов цикла. Для определения этого количества (n) используем формулу

$n = (i_{кон} – i_{нач})/h+1$.

В данном случае при $i_{кон}=31$, $i_{нач}=5$, $h=2$ получаем $n = 14$. Тогда программа будет иметь вид

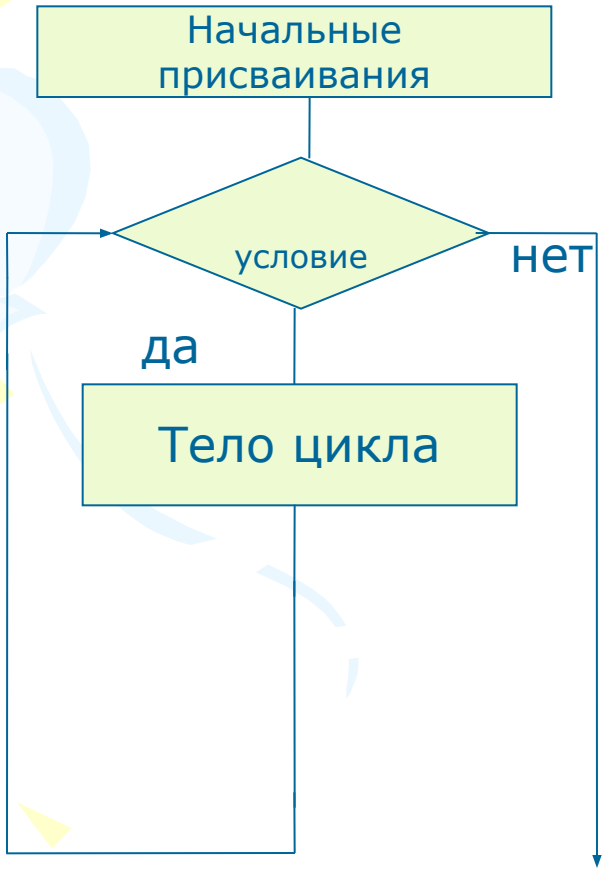
```
double s = 0;
for (int a = 5; a <= 31; a = a + 2)
{
    s = s + a / (a + 3.0);
}
Console.WriteLine("s = {0}", s);
```

Заметим, что управляющая переменная цикла не используется в вычислениях, а служит только для организации цикла.


Flash05.swf

# Циклы по условию

- количество повторений цикла **неизвестно** до начала выполнения цикла и в ряде случаев является искомой величиной при решении задачи.




- **Оператор `while`**

While (*условие*)

{

*Операторы*

}

- Здесь *условие* - условие продолжения цикла.
- *Условие* в простейшем случае имеет вид отношения. Например,

```
while (a>= 0)
...
```

# Оператор цикла while

**Оператор while** реализует цикл по условию с проверкой условия до первого прохождения цикла (цикл с предусловием). Общий вид цикла

```
While (условие)
{
  Операторы
}
```

Операторы выполняются, пока условие имеет значение true. Например:

```
int i = 1;
while (i<6)
{
   Console.WriteLine(i);
   i++;
}
```

В последовательные строки выводятся числа от 1 до 5.

# Операторы цикла do-while

**Операторы цикла do-while** реализуют цикл по условию с проверкой условия после первого прохождения цикла (цикл с постусловием). Общий вид цикла

```
do
{
  Операторы
}
while (условие);
```

Операторы выполняются, пока условие имеет значение true. Например:

```
int i = 1;
      do
      {
         Console.WriteLine(i);
         i++;
      }
      while (i < 6);
```

Результат выполнения программы будет тот же, что и в предыдущем примере.

Цикл можно прервать оператором *break*. Для перехода непосредственно к оператору вычисления выражения while (проверке условия) используется оператор *continue*.

# Пример

**Определить количество (n) членов арифметической прогрессии**

$s = a + (a + h) + (a + 2h) + .... + (a + nh)$,

**сумма которых наиболее близка к заданному числу *p*, но не превосходит его**.

На каждом шаге алгоритма нужно добавлять к сумме s очередной член

$m = a + ih$ ($s = s + m$), $i = 1, 2, ...,$

но при этом перед каждым прибавлением очередного члена проверять условие

$s <= p$.

Как только в первый раз условие не будет выполнено, т.е. будет $s > p$, необходимо выйти из цикла. Тогда **предпоследнее** значение номера слагаемого (n) и будет результатом.

Программа для решения задачи при
$a = 2$, $h = 3$, $p = 41$
имеет вид

```
int s = 0, n = 0, m;
const int a = 2, h = 3, p = 41;
while (s <= p)
{
    m = a + n * h;
    s = s + m;
    n = n + 1;
}
//вычитается 1, прибавленная
   после
//последнего изменения суммы.
n = n - 1;
Console.WriteLine("{0:d}", n);
```


Flash06.swf

- Второй вариант программы использует оператор цикла с проверкой условия после первого прохождения цикла.
- int s = 0, n = 0, m;
- const int a = 2, h = 3, p = 41;
- do
- {
- m = a + n*h;
- s = s + m;
- n = n + 1;
- }while(s <= p);
- n = n - 1;
- Console.WriteLine("{0:d}", n);

# Вычисление рядов

- ***Ряд*** – это бесконечная сумма вида

$u_1+u_2+u_3+\ldots+u_n+\ldots$

Или, короче, $\sum_{n=1}^{\infty} u_n$

- Ряды являются важнейшим средством вычисления чисел и функций. Например,

$$e = 1 + \frac{1}{1!} + \frac{1}{2!} + \frac{1}{3!} + \ldots$$ (числовой ряд).

Значение функции *sin x* может быть вычислено с помощью ряда

$$\sin x = x - \frac{x^3}{3!} + \frac{x^5}{5!} + \ldots + (-1)^n \frac{x^{2n+1}}{(2n+1)!} + \ldots$$ (функциональный ряд).

- При помощи рядов вычисляются практически все арифметические стандартные функции.
- Вычисление рядов требует организации циклов по счетчику, циклов по условию, вывода рекуррентных соотношений, использования вложенных циклов при решении задач табулирования функций, задаваемых рядами.

# Пример

Вычислить сумму $S = \cos x + \frac{\cos 2x}{2} + \ldots + \frac{\cos nx}{n} + \ldots$ при $x = 0.5$.

Суммирование прекратить, когда очередной член суммы по абсолютной величине будет меньше заданного ε = 0.0001.

```
double x = 0.5;
const double eps = 0.0001;
double s = 0, a;
int n = 1;
do
{
    a = Math.Cos(n*x) / n;
    s = s + a;
    n = n + 1;
} while (Math.Abs(a) > eps);
Console.WriteLine("Сумма равна {0:f4}", s);
Console.ReadKey();
```


Flash06.swf

# Пример

- **Вычислить сумму** $$s = \sum_{i=0}^{12} \frac{(2x)^i}{i!}$$

при *x*, изменяющемся в пределах от 0.1 до 1 с шагом 0.05.

## Вычисление суммы при заданном значении *x*

- Член суммы вычисляем рекуррентно. Для вывода рекуррентной формулы выпишем разделим *i*-ый член на (*i*–1)-ый

$$a_{i-1} = \frac{(2x)^{i-1}}{(i-1)!} \qquad a_i = \frac{(2x)^i}{i!} \qquad \frac{a_i}{a_{i-1}} = \frac{2x}{i}$$

Вычисление суммы для фиксированного значения x (например, x=0.2) может быть осуществлено следующим образом:

```
double s, a, x = 0.1;
s = 1; a = 1;
for (int i = 1; i <= 12; i++)
{
   a = a * 2 * x / i;
   s = s + a;
}
Console.WriteLine("{0:f4}    {1:f4}", x, s);
```

Эта последовательность операторов должна быть выполнена в цикле по x.

Flash07.swf

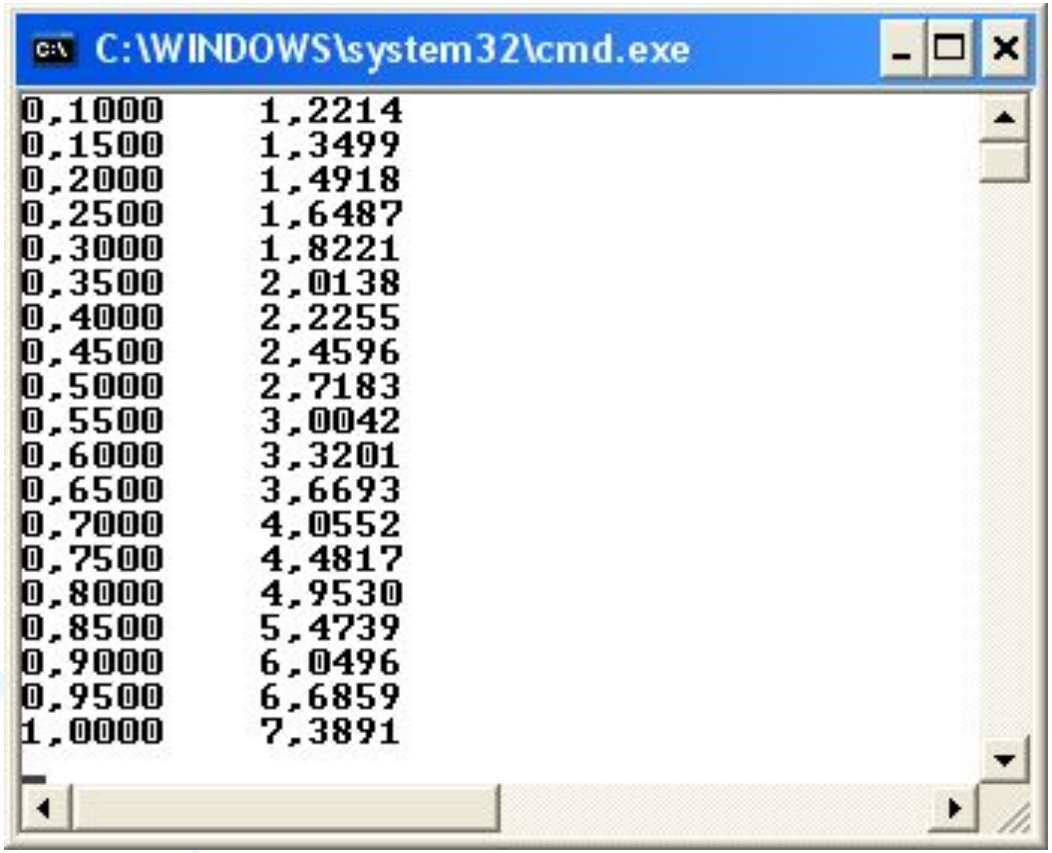
# **Вычисление суммы при изменении *x* в пределах** от 0.1 до 1 с шагом 0.05

```
double s, a, x;
double xh = 0.1, xk = 1.0001, h = 0.05;
int n = (int)((xk - xh) / h + 1);
x = xh;
for (int j = 1; j <= n; j++)
{
    s = 1; a = 1;
    for (int i = 1; i <= 12; i++)
    {
        a = a * 2 * x / i;
        s = s + a;
    }
    Console.WriteLine("{0:f4}    {1:f4}", x, s);
    x = x + h;
}
```

Такая структура программы, когда цикл выполняется внутри другого цикла, называется **вложенными циклами**.

Flash07.swf

# Результат


C:\WINDOWS\system32\cmd.exe
0,1000 1,2214
0,1500 1,3499
0,2000 1,4918
0,2500 1,6487
0,3000 1,8221
0,3500 2,0138
0,4000 2,2255
0,4500 2,4596
0,5000 2,7183
0,5500 3,0042
0,6000 3,3201
0,6500 3,6693
0,7000 4,0552
0,7500 4,4817
0,8000 4,9530
0,8500 5,4739
0,9000 6,0496
0,9500 6,6859
1,0000 7,3891

# Пример.

Вычислить сумму $$S = 1 + \frac{x^2}{2} - \frac{x^4}{8} + \cdots + (-1)^{i-1}\frac{(2i-1)x^{2i}}{(2i)!} + \ldots$$

для значений $x$, изменяющихся в пределах от 0.2 до 1 с шагом 0.2. Суммирование прекращать, когда очередной член суммы по абсолютной величине станет меньше $\varepsilon = 0.0001$. (Эта сумма является разложением в ряд функции $\cos x + x\sin x$).

## Разработка алгоритма

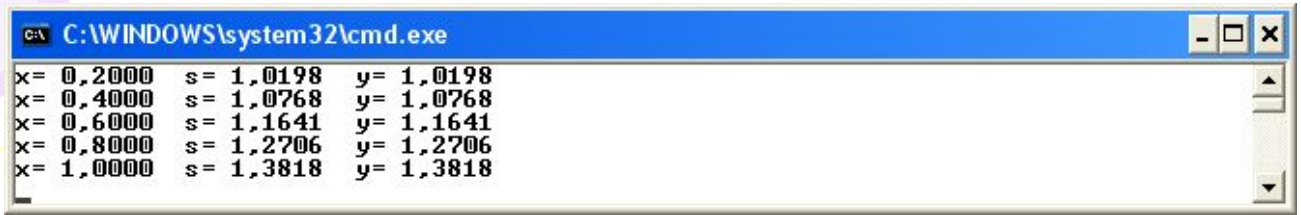
- Задача сводится к организации вложенных циклов. Внешний цикл по счетчику обеспечивает изменение $x$. Во внутреннем цикле по условию осуществляется вычисление суммы.
- Член суммы $a_i$ имеет более сложный вид, чем в предыдущем примере. Его целесообразно представить в виде двух сомножителей:
- $a_n = c_n(2n - 1)$, где $$c_n = (-1)^{n-1}\frac{x^{2n}}{(2n)!}$$

будем вычислять по рекуррентной формуле: $$c_n = -c_{n-1}\frac{x^2}{((2n-1)2n)}$$

- Число значений $x$ на отрезке от 0.2 до 1 с шагом 0.2 равно 5. В программе для контроля при каждом значении $x$ вычисляется также функция, которая приближенно может быть представлена в виде указанной суммы.

# Пример. Программа

```
const double xh = 0.2, h = 0.2, eps = 0.0001;
double a, x, s, y, c;
int n = 5, i;
x = xh;
for (int j = 1; j <= n; j++)
{
   s = 1; c = -1; i = 1;
   do
   {
      c = -c * x * x / ((2 * i - 1) * 2 * i);
      a = c * (2 * i - 1);
      s = s + a;
      i = i + 1;
   } while (Math.Abs(a) >= eps);
   y = Math.Cos(x) + x * Math.Sin(x);
   Console.WriteLine("x= {0:f4}  s= {1:f4}  y= {2:f4}", x, s, y);
   x = x + h;
}
```



# Вопросы для самопроверки

1. Что такое цикл по счетчику? Операторы цикла for.
2. Какие данные необходимы для организации цикла по счетчику? Что такое управляющая переменная цикла?
3. Циклы по условию и их организация. Как организовать проверку условия выхода из цикла до первого прохождения цикла, после первого прохождения цикла? Различия между циклами с предусловием и постусловием.
4. Операторы выхода из цикла. В каких случаях они используются?
5. Типовые алгоритмы циклической структуры: вычисление суммы *n* слагаемых, вычисление произведения *n* сомножителей, вычисление факториала, табулирование функции.
6. Что такое рекуррентное соотношение?
7. Вычисление суммы с использованием рекуррентных соотношений.
8. Вложенные циклы.

Стр.47, №№1-16; №№17-25 (ряды)