

Лабораторная работа № 1.1

«Алгоритмы вычисления сумм числовых рядов»

В данной лабораторной работе рассматриваются способы (алгоритмы) вычисления сумм S числовых рядов $\sum a_k$, когда количество слагаемых является либо заданным конечным числом, например $S = \sum_{k=1}^N a_k$, либо когда ряд является бесконечным, т.е. $S = \sum_{k=1}^{\infty} a_k$.

Задача вычисления конечной суммы числового ряда $S = \sum_{k=1}^N a_k$, очевидно, является более простой, чем задача вычисления суммы аналогичного бесконечного числового ряда $S = \sum_{k=1}^{\infty} a_k$.

Причина кроется в том, что для бесконечного числового ряда мы не можем реально выполнить сложение всех членов этого ряда. Бесконечный числовой ряд, в конечном итоге, все равно будет заменяться каким-то конечным числовым рядом. Вопрос лишь в том, как это повлияет на точность получаемого итогового результата.

Точность результата (погрешность вычислений), как правило, задается в качестве ключевого параметра вычислительного процесса, ограничивающего его продолжительность.

Цель лабораторной работы

Разработать алгоритмы и на их основе реализовать программные методы вычисления сумм числовых рядов, когда количество слагаемых членов ряда конечно и когда оно предполагается бесконечным. Указанные методы следует разместить в специальном классе **MAC_Series** проекта динамической библиотеки **MAC_DLL**.

Отладить консольное приложение (отдельный проект), предназначенное для вычисления «тестовых» сумм двух заданных числовых рядов.

Подтверждение Вами тестовых результатов, что приводятся ниже в данном руководстве, свидетельствует о завершении первого этапа данной лабораторной работы.

Второй этап – проведение вычислений на основе индивидуального варианта задания.

Студент заполняет результатами соответствующую форму, что выдается ему преподавателем сразу после подтверждения тестовых результатов. Заполненная форма сдается преподавателю по окончании лабораторного занятия для проверки. Все разработанные программные компоненты сохраняются студентом до конца учебного семестра.

Порядок выполнения задания лабораторной работы на языке C#

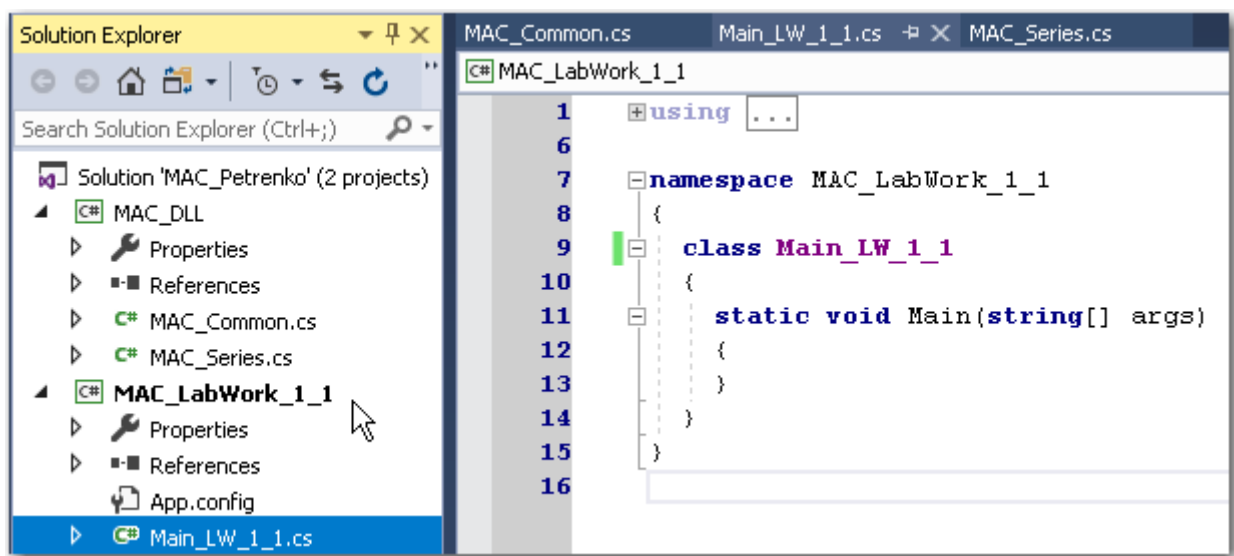
Стартуем среду разработки приложений **MS Visual Studio 2017**.

В имеющемся рабочем пространстве **MAC_Petrenko** генерируем новый проект консольного приложения **MAC_LabWork_1_1**.

Выбираем тип конфигурации рабочего проекта – **Debug**.

Для удобства сразу переименуем класс **Program** в класс **Main_LW_1_1**.

Это поможет Вам различать файлы проектов ***.cs** их по уникальным именам, когда подобных проектов в рабочем пространстве **MAC_Petrenko** будет уже много.



Первый этап лабораторной работы будет заключаться в создании универсального метода, выполняющего вычисление конечной суммы членов некоторого числового ряда.

Указанная задача имеет следующую математическую формулировку:

$$S = \sum_{k=k_I}^{k_L} a_k, \quad (1.1.1)$$

где S – искомое значение суммы, (*sum*),
 k – индекс суммирования, (*index*), инкремент индекса суммирования равен 1,
 k_I – начальное значение индекса, (*initial index*),
 k_L – конечное значение индекса, (*last index*),
 a_k – член числового ряда, функционально зависящий от индекса k , (*member*),
 например, $a_k = \frac{k+1}{(k+2)^3}$.

Разрабатываемый метод разместим в динамической библиотеке **MAC_DLL** в отдельном классе – **MAC_Series** (переименуем уже имеющийся класс **Class1**):

```

1  using ...
6
7  namespace MAC_DLL
8  {
9      public class MAC_Series
10     {
11         //-----
12         //  Вычисление суммы членов числового ряда Members,
13         //  начиная от члена с индексом Initial_Index и
14         //  заканчивая членом с индексом Last_Index
15
16         public static double Sum_of_Number_Series
17         (int Initial_Index, int Last_Index, Member_of_Numeric_Series Members)
18         {
19             double global_sum = 0.0;
20             for (int k = Initial_Index; k <= Last_Index; k++)
21             {
22                 global_sum += Members(k);
23             }
24             return global_sum;
25         }
26     }
27 }
    
```

Суть алгоритма, реализованного в методе **Sum_of_Number_Series()**, очевидна из его определения (1.1.1).

В отдельном файле – **MAC_Common.cs** – в пространстве имен библиотеки **MAC_DLL** даём определение используемому делегату **Member_of_Numeric_Series()**:

```

MAC_Common.cs  X Main_LW_1_1.cs  MAC_Series.cs
[MAC_DLL]
1  using ...
6
7  namespace MAC_DLL
8  {
9      # region <--- Делегаты --->
10
11     public delegate double Member_of_Numeric_Series(int Index);
12
13     # endregion <--- Делегаты --->
14 }
    
```

Пусть в рамках задания на лабораторную работу нам предлагаются следующие бесконечные числовые ряды, которые будем использовать в качестве «тестовых»:

$$S_1 = \sum_{k=1}^{\infty} \frac{1}{(2k+1)^2} = \frac{\pi^2}{8} - 1 \quad \text{и} \quad S_2 = \sum_{k=0}^{\infty} \frac{(-1)^k}{(2k+1)^3} = \frac{\pi^3}{32}. \quad (1.1.2)$$

Для тестовых рядов (1.1.2) уже посчитаны точные значения сумм, когда число слагаемых членов в этих рядах бесконечно.

Обратите внимание на то, что второй числовой ряд, в отличие от первого, является знакопеременным. Слагаемые второго числового ряда поочередно меняют знак.

В классе **Main_LW_1_1** создадим коды статических методов, соответствующих сигнатуре делегата **Member_of_Numeric_Series()**, и реализующих вычисления членов тестовых числовых рядов (1.1.2):

$$a_k = \frac{1}{(2k+1)^2} \quad \text{и} \quad b_k = \frac{(-1)^k}{(2k+1)^3}. \quad (1.1.3)$$

```

1  using ...
6
7  namespace MAC_LabWork_1_1
8  {
9      class Main_LW_1_1
10     {
11     public:
12         static void Main(string[] args) ...
13
14         static double My_ak(int k)
15         {
16             return 1.0 / (2.0 * k + 1.0) / (2.0 * k + 1.0);
17         }
18
19         static double My_bk(int k)
20         {
21             double b = (2.0 * k + 1.0);
22             if ((k % 2) == 0) return 1.0 / b / b / b; else return -1.0 / b / b / b;
23         }
24     }
25 }
```

Теперь (известным Вам способом) следует добавить динамическую библиотеку **MAC_DLL** в раздел **References** проекта **MAC_LabWork_1_1**.

Это позволит упоминать библиотеку **MAC_DLL** и ее открытые члены (классы) в директиве **using** в разрабатываемом консольном приложении **MAC_LabWork_1_1**.

Теперь все готово для разработки первой части основного приложения данной лабораторной работы, вычисляющей, например, конечные тестовые суммы (1.1.2):

$$S_1 = \sum_{k=1}^N \frac{1}{(2k+1)^2} \quad \text{и} \quad S_2 = \sum_{k=0}^N \frac{(-1)^k}{(2k+1)^3}, \quad (1.1.4)$$

когда задано $N = 10\,000$ – верхняя граница (наибольшее значение) индекса суммирования.

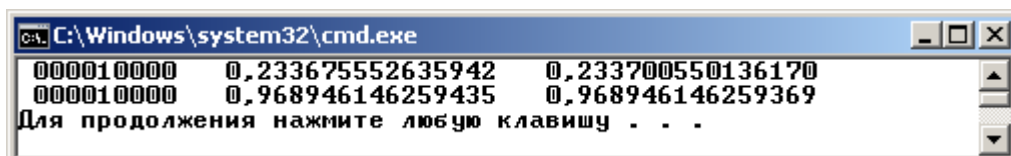
```

1  using System;
2  using System.Collections.Generic;
3  using System.Linq;
4  using System.Text;
5  using System.Threading.Tasks;
6  using CLS = MAC_DLL.MAC_Series;
7
8  namespace MAC_LabWork_1_1
9  {
10     class Main_LW_1_1
11     {
12         static void Main(string[] args)
13         {
14             double True_Sum1 = Math.PI * Math.PI / 8.0 - 1.0;
15             double True_Sum2 = Math.Pow(Math.PI, 3.0) / 32.0;
16             string format = " {0:D9}    {1:F15}    {2:F15}";
17
18             int N = 10000;
19
20             double S1_N = CLS.Sum_of_Number_Series(1, N, My_ak);
21             Console.WriteLine(format, N, S1_N, True_Sum1);
22
23             double S2_N = CLS.Sum_of_Number_Series(0, N, My_bk);
24             Console.WriteLine(format, N, S2_N, True_Sum2);
25         }
26
27         static double My_ak(int k) ...
31         static double My_bk(int k) ...
36     }
37 }

```

Для контроля получаемого результата одновременно вычисляются **точные** значения сумм (1.1.2) для бесконечных рядов.

Очевидно, что с увеличением значения параметра N , разница соответствующих значений конечных и бесконечных сумм должна уменьшаться.



```

C:\Windows\system32\cmd.exe
000010000  0.233675552635942  0.233700550136170
000010000  0.968946146259435  0.968946146259369
Для продолжения нажмите любую клавишу . . .

```

Имеем достаточно убедительный результат – совпадение определенного количества значащих цифр в обеих суммах.

Таким образом, алгоритм вычисления суммы конечного числа слагаемых для заданного числового ряда, реализованный в статическом методе `Sum_of_Number_Series()`, можно считать отлаженным.

Второй этап лабораторной работы будет заключаться в создании универсального метода, выполняющего **приближенное** вычисление «бесконечной» суммы членов некоторого числового ряда с заданной **абсолютной** погрешностью ε .

Указанная задача имеет следующую математическую формулировку:

$$S = \sum_{k=k_I}^{\infty} a_k = \sum_{k=k_I}^{k_F} a_k + \sum_{k=k_F+1}^{\infty} a_k = S(k_F) + \tilde{S} \approx S(k_F), \quad |\tilde{S}| < \varepsilon, \quad (1.1.5)$$

где $S(k_F)$ – искомое приближенное значение суммы, (*sum*),

k – индекс суммирования, (*index*), инкремент индекса суммирования, равный 1,

k_I – начальное значение индекса, (*initial index*),

k_F – конечное значение индекса, (*final index*), определяемое в ходе вычислений,

\tilde{S} – сумма «отбрасываемой» части бесконечного числового ряда, для которой предполагается выполнение неравенства $|\tilde{S}| < \varepsilon$.

Данная задача, вообще говоря, не может быть признана корректной математически, поскольку для произвольного числового ряда мы не можем программным путем «оценить» порядок суммы \tilde{S} «отбрасываемой» части ряда по отношению к итоговому результату $S(k_F)$.

Можно предложить некоторый алгоритм, который будет давать удовлетворительный результат для хорошо сходящихся числовых рядов, и будет давать определенную погрешность для плохо сходящихся числовых рядов.

Наша основная цель – правильно запрограммировать предлагаемый числовой алгоритм и затем проверить его на тестовых примерах. Итак, суть предлагаемого алгоритма следующая.

На первом этапе вычисления суммы ряда выполняется подсчет суммы s_0 членов от начального члена a_{k_I} , до того члена ряда a_n , модуль которого будет меньше заданной

погрешности вычислений, т.е. $s_0 = \sum_{k=k_I}^n a_k$, где $|a_n| < \varepsilon$.

Таким образом, сумма s_0 будет вычислена от начального члена ряда a_{k_I} до члена a_n включительно, а общее количество членов в этой сумме составит величину $N = n - k_I + 1$.

Чем медленнее (хуже) числовой ряд сходится, тем большей окажется эта величина N .

Например, если $k_I = 0$, а $|a_5| < \varepsilon$, т.е. $n = 5$, то сумма s_0 будет вычислена на основе первых $N = 5 - 0 + 1 = 6$ – шести первых членов ряда: a_0, a_1, \dots, a_5 .

Второй этап алгоритма.

Следующая вычисляемая сумма – s_1 – это сумма из N последующих членов данного числового ряда.

При этом первым слагаемым этой суммы будет член a_{n+1} , а последним – a_{n+N} .

В нашем «примере» это будут члены от a_6 и до a_{11} включительно.

Полученная сумма s_1 сразу прибавляется к уже имеющемуся значению s_0 .

Если выполняется неравенство $|s_1| < \varepsilon$, то данный процесс заканчивается и результатом будет признана «обновленная» сумма s_0 .

Если же имеет место $|s_1| \geq \varepsilon$, то вычисляется **новая** сумма s_1 членов заданного ряда от a_{n+N+1} до a_{n+2N} .

И так далее... до выполнения условия $|s_1| < \varepsilon$.

Указанный алгоритм оформим в виде нового метода `Sum_of_Number_Series_A()`, размещенного в классе **MAC_Series** разрабатываемой динамической библиотеки **MAC_DLL**.

```

1  using System;
2  using System.Collections.Generic;
3  using System.Linq;
4  using System.Text;
5  using System.Threading.Tasks;
6
7  namespace MAC_DLL
8  {
9      public class MAC_Series
10     {
11         // ----- ...
12
13         public static double Sum_of_Number_Series
14             (int Initial_Index, int Last_Index, Member_of_Numeric_Series Members) ...
15
16         // -----
17         // Вычисление суммы членов числового ряда Members, начиная от члена
18         // с индексом Initial_Index, с заданной абсолютной погрешностью Eps.
19         // Параметр Final_Index возвращает индекс последнего члена ряда,
20         // использованного при подсчете общей суммы ряда.
21
22         public static double Sum_of_Number_Series_A
23             (int Initial_Index, double Eps, Member_of_Numeric_Series Members,
24              ref int Final_Index)
25         {
26             double ak, sum_1, sum_0 = 0.0; int k = Initial_Index; bool flag;
27             do
28             {
29                 ak = Members(k); sum_0 += ak; flag = (Math.Abs(ak) >= Eps);
30                 if (flag) k++;
31             } while (flag);
32
33             int N = k - Initial_Index + 1; k++;
34             do
35             {
36                 sum_1 = Sum_of_Number_Series(k, k + N, Members);
37                 sum_0 += sum_1; flag = (Math.Abs(sum_1) >= Eps);
38                 if (flag) k = k + N + 1;
39             } while (flag);
40
41             Final_Index = k + N; return sum_0;
42         }
43     }
44 }

```

Примечание. При разработке части программного кода, реализующего второй этап алгоритма – этап вычисления сумм s_1 , мы воспользовались уже имеющимся в библиотеке **MAC_DLL** методом `Sum_of_Number_Series()` для вычисления конечных сумм.

Иногда приближенные вычисления требуют выполнять с заданной *относительной* погрешностью δ .

Это касается и задач вычисления сумм бесконечных числовых рядов.

Очевидно, что мы можем воспользоваться уже разработанным предыдущим алгоритмом, где вычисления производились с учетом абсолютной погрешности ε .

Первый этап нового алгоритма идентичен первому этапу предыдущего алгоритма, на котором проверяется выполнение условия $|a_n| < \delta$ и определяется значение N .

На втором этапе нового алгоритма циклические вычисления должны продолжаться по следующему правилу.

Если выполняется неравенство $\left| \frac{s_1}{s_0} \right| < \delta$, то данный итерационный процесс заканчивается и результатом будет признана «обновленная» сумма s_0 .

Имеется в виду то, что сумма ряда s_0 на каждой итерации должна «обновляться» путем прибавления к ней нового значения частичной суммы s_1 .

Если же имеет место $\left| \frac{s_1}{s_0} \right| \geq \delta$, то вычисляется новая сумма s_1 членов заданного числового ряда от a_{n+N+1} до a_{n+2N} .

И так далее... до выполнения условия $\left| \frac{s_1}{s_0} \right| < \delta$.

Данный алгоритм (с учетом относительной погрешности) вычисления суммы числового ряда оформим в виде нового метода `Sum_of_Number_Series_D()`, и также разместим в классе **MAC_Series** разрабатываемой динамической библиотеки **MAC_DLL**.

```

55 //-----
56 // Вычисление суммы членов числового ряда Members, начиная от члена
57 // с индексом Initial_Index, с заданной относительной погрешностью Delta.
58 // Параметр Final_Index содержит индекс последнего члена ряда,
59 // использованного при подсчете суммы ряда.
60
61 public static double Sum_of_Number_Series_D
62 (int Initial_Index, double Delta, Member_of_Numeric_Series Members,
63 ref int Final_Index)...
```

Эту часть работы Вы выполняете самостоятельно.

Тестирование разработанных методов (алгоритмов суммирования) будем выполнять при помощи единого консольного приложения **MAC_LabWork_1_1**:

```

1  using System;
2  using CLS = MAC_DLL.MAC_Series;
3  // using System.Collections.Generic; ...
7
8  namespace MAC_LabWork_1_1
9  {
10     class Main_LW_1_1
11     {
12         static void Main(string[] args)
13         {
14             double True_Sum1 = Math.PI * Math.PI / 8.0 - 1.0;
15             double True_Sum2 = Math.Pow(Math.PI, 3.0) / 32.0;
16             string format = " {0:D9}    {1:F15}    {2:F15}";
17             int kF = 0, N = 10000; double Eps = 1.0E-9, Dlt = 1.0E-8;
18
19             double S1_N = CLS.Sum_of_Number_Series(1, N, My_ak);
20             Console.WriteLine(format, N, S1_N, True_Sum1);
21
22             double S1_E = CLS.Sum_of_Number_Series_A(1, Eps, My_ak, ref kF);
23             Console.WriteLine(format, kF, S1_E, True_Sum1); Console.WriteLine();
24             double S1_D = CLS.Sum_of_Number_Series_D(1, Dlt, My_ak, ref kF);
25             Console.WriteLine(format, kF, S1_D, True_Sum1); Console.WriteLine();
26
27             double S2_N = CLS.Sum_of_Number_Series(0, N, My_bk);
28             Console.WriteLine(format, N, S2_N, True_Sum2);
29             double S2_E = CLS.Sum_of_Number_Series_A(0, Eps, My_bk, ref kF);
30             Console.WriteLine(format, kF, S2_E, True_Sum2); Console.WriteLine();
31         }
32
33         static double My_ak(int k) ...
37         static double My_bk(int k) ...
42     }
43 }

```

Результатом выполнения функции **Main()** должна быть следующая таблица:

```

C:\Windows\system32\cmd.exe
000010000    0,233675552635942    0,233700550136170
002008123    0,233700425641866    0,233700550136170

000735146    0,233700210068130    0,233700550136170

000010000    0,968946146259435    0,968946146259369
000001002    0,968946146321312    0,968946146259369

Для продолжения нажмите любую клавишу . . .

```

После выполнения расчетов с «тестовыми» суммами (1.1.2) и (1.1.4) Вы должны реализовать вычисления в соответствии с индивидуальным вариантом данной Лабораторной работы, который выдается Вам на специальном бланке.