ЛАБОРАТОРНАЯ РАБОТА № 2

Ввод-вывод информации, с использованием файлов. Форматирование значений данных.

1. Краткие теоретические сведения

Вывод, производимый методами `System.Console.Write()` и `System.Console.WriteLine()`, можно форматировать. Форматирование позволяет указывать формат целых чисел, чисел с плавающей точкой и других типов данных.

Управление форматом числовых данных

Пусть в программе определена переменная типа `int` с именем `value`:

```
int value = 250;
```

До этого момента переменные выводились следующим образом:

```
System.Console.WriteLine ("value  =" + value);
```

Результат вывода: `value = 250`

Можно вывести значение `value`, используя требуемое число позиций (например 5):

```
System.Console.WriteLine("value = {0, 5}", value);
```

Первое число в фигурных скобках означает номер переменной – это `0`, что соответствует первой переменной `value` в списке метода `System.Console.WriteLine()`. Второе число в фигурных скобках означает количество позиций, отведенное для отображения переменной. В данном примере оно равно `5`. При выводе переменной длина ее представления будет дополнена пробелами сле-

ва. Если количество позиций меньше чем число знаков переменной, то оно будет выведено без форматирования.

Можно задать форматирование для вывода каждой переменной:

```
int a = -12;
int b = 20;
System.Console.WriteLine("a = {0,  4}, b =  {1,  3}",
a,  b);
```

Результат вывода: `a = -12, b = 20`

Форматированный вывод чисел с плавающей точкой немного более сложный. Предположим, определена переменная типа double с именем `myDouble`: `double myDouble = 1234.56789;`

Следующий пример выводит значения `myDouble`, отведя под него десять знакомест, и округлив его до трех цифр после запятой:

```
System.Console.WriteLine("myDouble  =  {0,    10:f3}";
myDouble);
```

Символы `f3` в этом примере означают, что значение выводится как число с плавающей точкой (символ `f`) , в дробной части будет выведено три цифры.

Точно такое же форматирование можно применять для типов `float` и `decimal`. Например:

```
float myFloat = 1234.56789f;
System.Console.Write("myFloat   =      {0,   10:f3}",
myFloat);
decimal myDecimal = 1234.56789m;
System.Console.Write("myDecimal  =     {0,   10:f3}",
myDecimal);
```

Результат вывода:

```
myFloat = 1234.568;
myDecimal 1234.568;
```

В списке аргументов методов `WriteLine` или `Write` задается строка вида `{n, w: спецификатор k}` – где `n` определяет номер идентификатора из списка аргументов метода `WriteLine`, *спецификатор* – определяет формат данных, `w` – целая константа без знака, задает количество символов (длину поля), а `k` – количество позиций для дробной части значения идентификатора.

Для каждого типа данных существует своя форма представления. Данные сведены в таблицу 2.1.

Таблица 2.1

Тип данных		Форма
Числовые	Целые	`W`
	Вещественные с фиксированной точностью	`W:Fk`
	Вещественные в экспоненциальном формате	`W:Ek`
Логические		`W`
Символьные		`W`

Символы форматирования `F`, `E` (другие символы форматирования приведены в табл. 2.2) – определяют тип и характеристики объектов ввода-вывода. Параметр `w` – целая константа без знака, задает количество символов (длину поля), отводимых для ввода-

вывода объекта. Параметр k – целая константа без знака определяет для числовых данных:

- количество позиций, для цифр в дробной части числа (форма F);
- количество позиций для цифр, в дробной части мантиссы числа (форма E или G).

В качестве спецификаторов могут использоваться следующие значения:

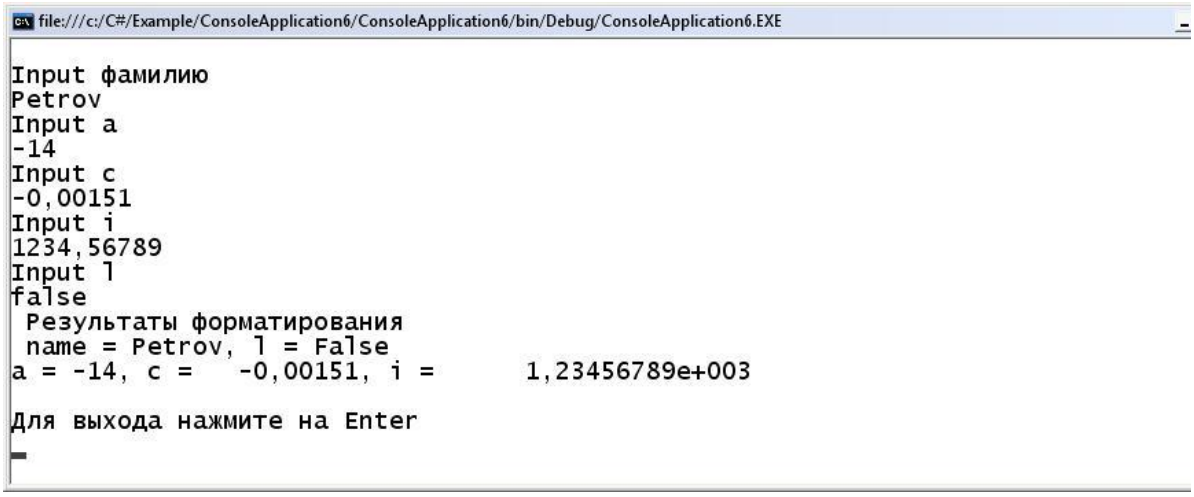
Таблица 2.2

Символ	Формат	Значение
C или c	Денежный. По умолчанию ставит знак p. Изменить его можно с помощь объекта NumberFormatInfo	Задается количество десятичных разрядов.
D или d	Целочисленный (используется только с целыми числами)	Задается минимальное количество цифр. При необходимости результат дополняется начальными нулями
E или e	Экспоненциальное представление чисел	Задается количество символов после запятой. По умолчанию используется 6
F или f	Представление чисел с фиксированной точкой	Задается количество символов после запятой
G или g	Общий формат (или экспоненциальный, или с фиксированной точкой)	Задается количество символов после запятой. По умолчанию выводится целая часть

N или n	Стандартное форматирование с использованием запятых и пробелов в качестве разделителей между разрядами	Задается количество символов после запятой. По умолчанию – 2, если число целое, то ставятся нули
X или x	Шестнадцатеричный формат	
P или p	Процентный	

Пример 1. Форматированный вывод данных различного типа.

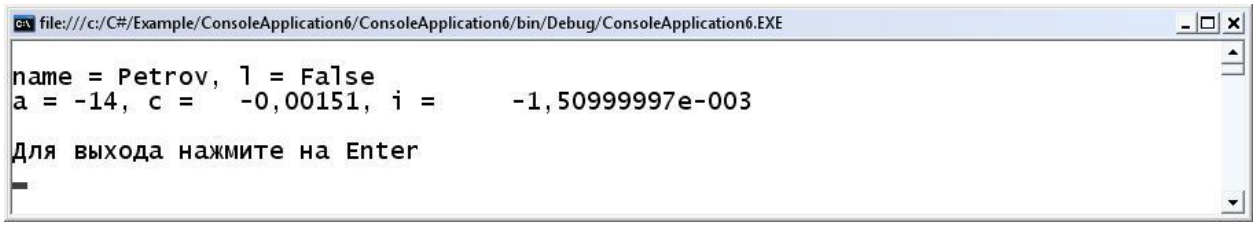
```
public static void Main()
{
    int a =  -14;
    float c = -0.00151f;
    double i = 1234.56789;
    bool l=false;
    string name="Petrov";
    System.Console.WriteLine("name = {0, 6}, l =
    {1, 4}", name,l);
    System.Console.WriteLine("a ={0, 4}, c =
    {1,10:f5}, i = {1,20:e8}", a,c,i);
    System.Console.WriteLine(" ");
    System.Console.WriteLine("Для выхода нажмите
    на Enter");
    System.Console.ReadLine();
}
```



Пример 2. Ввод в диалоге и форматированный вывод данных различного типа.

```
public static void Main()
{
    int a;      // =  -14;
    float c;    // = -0.00151f;
    double i;   // = 1234.56789;
    bool l;     // = false;
    string name; //="Petrov";
    Console.WriteLine("Input фамилию ");
    name = Console.ReadLine();
    Console.WriteLine("Input a");
    a = Convert.ToInt32(Console.ReadLine());
    Console.WriteLine("Input c");
    c = Convert.ToSingle(Console.ReadLine());
    Console.WriteLine("Input i");
    i = Convert.ToDouble(Console.ReadLine());
    Console.WriteLine("Input l");
    l = Convert.ToBoolean(Console.ReadLine());
```

```
    System.Console.WriteLine(" Результаты
    форматирования \n name = {0, 6}, l = {1, 4}",
    name,l);
    System.Console.WriteLine("a ={0, 4}, c =
    {1,10:f5}, i = {2,20:e8}", a,c,i);
    System.Console.WriteLine(" ");
    System.Console.WriteLine("Для выхода нажмите
    на Enter");
    System.Console.ReadLine();
}
```



Организация ввода вывода с использованием файлов

C#-программы выполняют операции ввода-вывода посредством потоков, которые построены на иерархии классов. Поток (stream) – это абстракция, которая генерирует и принимает данные. С помощью потока можно читать данные из различных источников (клавиатура, файл) и записывать в различные источники (принтер, экран, файл). Несмотря на то, что потоки связываются с различными физическими устройствами, характер поведения всех потоков одинаков. Поэтому классы и методы ввода-вывода можно применить ко многим типам устройств.

На самом низком уровне иерархии потоков ввода-вывода находятся потоки, оперирующие байтами. Это объясняется тем, что многие устройства при выполнении операций ввода-вывода ориен-

тированы на байты. Однако для человека привычнее оперировать символами, поэтому разработаны символьные потоки, которые фактически представляют собой оболочки, выполняющие преобразование байтовых потоков в символьные и наоборот. Кроме этого, реализованы потоки для работы с `int`-, `double`-, `short`- значениями, которые также представляют оболочку для байтовых потоков, но работают не с самими значениями, а с их внутренним представлением в виде двоичных кодов.

Центральную часть потоковой C#-системы занимает класс `Stream` пространства имен `System.IO`. Класс `Stream` представляет байтовый поток и является базовым для всех остальных потоковых классов. Чтобы создать символьный поток нужно поместить объект класса `Stream` (например, `FileStream`) "внутрь" объекта класса `StreamWriter` или объекта класса `StreamReader`. В этом случае байтовый поток будет автоматически преобразовываться в символьный.

Класс StreamWriter предназначен для организации выходного символьного потока. Этот класс содержит несколько конструкторов. Так, например, создать экземпляр класса `StreamWriter` можно следующим образом:

```
StreamWriter fileOut = new StreamWriter(new
FileStream("text.txt", FileMode.Create,
FileAccess.Write));
```

Эта версия конструктора позволяет ограничить доступ только чтением или только записью:

```
FileStream(string filename, FileMode mode,
FileAccess how)
```

где:

1. параметры filename и mode имеют то же назначение, что и в предыдущей версии конструктора;
2. параметр how, определяет способ доступа к файлу и может принимать одно из значений, определенных перечислением FileAccess:

```
FileAccess.Read - только чтение;
FileAccess.Write - только запись;
FileAccess.ReadWrite - и чтение, и запись.
```

Другой вид конструктора позволяет открыть поток сразу через обращения к файлу:

```
StreamWriter(string name),
```

где параметр name определяет имя открываемого файла.

Например, обратиться к данному конструктору можно следующим образом:

```
StreamWriter fileOut = new
StreamWriter("c:\temp\t.txt");
```

И еще один вариант конструктора StreamWriter:

```
StreamWriter(string name, bool appendFlag),
```

где параметр name определяет имя открываемого файла; параметр appendFlag может принимать значение true – если нужно добавлять данные в конец файла, или false – если файл необходимо перезаписать.

Например:

```
StreamWriter fileOut=new StreamWriter("t.txt", true);
```

Теперь для записи данных в поток fileOut можно обратиться к методу WriteLine. Это можно сделать следующим образом:

```
fileOut.WriteLine("test");
```

В данном случае в конец файла t.txt будет дописано слово test.

Класс StreamReader предназначен для организации входного символьного потока. Один из его конструкторов выглядит следующим образом:

```
StreamReader(Stream stream),
```

где параметр `stream` определяет имя уже открытого байтового потока. Этот конструктор генерирует исключение типа `ArgumentException`, если поток `stream` не открыт для ввода.

Например, создать экземпляр класса `StreamWriter` можно следующим образом:

```
StreamReader fileIn = new StreamReader(new
FileStream("text.txt", FileMode.Open,
FileAccess.Read));
```

Как и в случае с классом `StreamWriter` у класса `StreamReader` есть и другой вид конструктора, который позволяет открыть файл напрямую:

```
StreamReader (string name);
```

где параметр `name` определяет имя открываемого файла.

Обратиться к данному конструктору можно следующим образом:

```
StreamReader fileIn=new StreamReader
("z:\temp\t.txt");
```

В C# символы реализуются кодировкой `Unicode`. Для того, чтобы можно было обрабатывать текстовые файлы, содержащие русский символы, созданные, например, в Блокноте, рекомендуется вызывать следующий вид конструктора `StreamReader`:

```
StreamReader fileIn=new StreamReader
("z:\temp\t.txt", Encoding.GetEncoding(1251));
```

Параметр `Encoding.GetEncoding(1251)` говорит о том, что будет выполняться преобразование из кода Windows-1251 (одна из

модификаций кода ASCII, содержащая русские символы) в Unicode. Encoding.GetEncoding(1251) реализован в пространстве имен System.Text.

Теперь для чтения данных из потока fileIn можно воспользоваться методом ReadLine. При этом если будет достигнут конец файла, то метод ReadLine вернет значение null.

По завершении работы с файлом его необходимо закрыть. Для этого достаточно вызвать метод Close(). При закрытии файла освобождаются системные ресурсы, ранее выделенные для этого файла, что дает возможность использовать их для работы с другими файлами.

Рассмотрим пример, в котором данные из одного файла считываются программой расчета функции и результаты помещаются в другой файл в заданной форме с использованием классов StreamWriter и StreamReader.

Пример 3. Ввод данных из файла и форматированный вывод данных различного типа в файл.

```
static void Main()
{
    string s;
    double x, y;
    StreamWriter f = new StreamWriter("out.txt");
    StreamReader f1 = new StreamReader("in.txt");
    f.WriteLine("        Таблица значений\n");
    metka: s = f1.ReadLine();
    if (s == null) goto metka1;
    x = Convert.ToDouble(s);
```

```
    y = Math.Sqrt(x * x / (2 + Math.Exp(4 *
    Math.Log(x))));
    f.WriteLine(" аргумент x = {0:F3} функция y =
    {1:e3} \n", x, y);
    goto metka;
    metka1: f.WriteLine("   Составил  Петров Иван
    {0} \n", s);
    f.Close();
    f1.Close();
}
```

Исходные данные файл in.txt

0,11
0,5
1

Результаты расчетов файл out.txt

```
        Таблица значений

аргумент x = 0,110 функция y = 7,778e-002

аргумент x = 0,500 функция y = 3,482e-001

аргумент x = 1,000 функция y = 5,774e-001

    Составил Петров Иван
```

2. Практическая часть

1) Составить программу для ввода в диалоге значений переменных A, I, C, L, Name и форматного вывода на экран монитора

введенных переменных (значения вводимых переменных даны в таблице 2.3).

2) Составить программу для вычисления и печати значений функции из таблицы 2.4. Вычислить 8 значений функции на заданном интервале. Исходные данные задать в файле LAB2.TXT. Результат поместить в файл вывода с именем LAB2.RES в заданной форме (таблица 2.5) .

Варианты задания

Таблица 2.3

Вариант	A	I	C	L	N
1	-14	-10^4	-0,00151	ложь	Фамилия
2	99,35	72	1995	истина	Имя
3	0,086	-19	4,025	ложь	Отчество
4	34	-6124	$3,2\times10^5$	истина	Фамилия
5	5,008	229	0,019	ложь	Имя
6	$3,5\times10^{-4}$	1989	-380,08	истина	Отчество
7	0,095	-1	1996	ложь	Фамилия
8	1,0074	10^2	107,7	истина	Имя
9	993,285	112000	$2,3\times10^{-4}$	ложь	Отчество
10	-2,1	444	10^3	истина	Фамилия
11	3,125	6006006	-13,24	ложь	Имя
12	-45,077	30	25×10^{12}	истина	отчество
13	12,97	1002	-999,7	ложь	фамилия
14	-0,09	2004	399,44	ложь	имя
15	-142	-10^4	-0,00151	истина	отчество
16	9,35	- 5072	19,95	ложь	фамилия
17	0, 86	-19726	4,025	истина	имя

18	34	-6	$3,2\times10^{3}$	ложь	отчество
19	5,008	-229	-0,019	истина	фамилия
20	$3,5\times10^{-4}$	1989	-380,08	ложь	имя
21	0,095	-12	1996	истина	отчество
22	1,0074	10^{2}	107,7	ложь	фамилия
23	993,285	112000	$2,3\times10^{-4}$	истина	имя
24	$-2,1\times10^{3}$	444	10^{-3}	ложь	отчество
25	3,125	6007007	-13,24	истина	фамилия
26	-45,07	123	25×10^{12}	ложь	имя
27	89,09	1000	999,002	истина	отчество
28	-99,78	11	-1,774	ложь	фамилия
29	7,99	-30077	1000	истина	истина
30	0,124	-100400	-9000	ложь	фамилия

Таблица 2.4

№	Функция	Контрольное значение		Интервал x		Вариант формы вывода
		x_*	y_*	x_{min}	x_{max}	
1	$y=\frac{1-x^2}{1+x^4}$	2	-0,176	-3	3	1
2	$y=\frac{\sin x}{x^2-1}$	1,57	0,406	-2	2	2
3	$y=\frac{2\pi}{x^2-\pi}$	3,14	0,935	-2	4	3
4	$y=4x+\frac{1}{x+1}$	1	4,5	0	2,5	4

5	$y=\frac{\sin^2 x}{x-1}$	1,57	1,75	1,5	5	1
6	$y=\frac{x^2-5x+4}{x^2+1}$	2	-0,4	-2	3	2
7	$y=2-\frac{e^{2x}+e^{-2x}}{e^2+e^{-2}}$	1	1	-1	1	3
8	$y=\frac{\sin x}{x^2+1}$	1,57	0,299	-2	2	4
9	$y=e^{-x}\sin(\frac{\pi x}{2})$	-1	-2,7	0	2,5	1
10	$y=\frac{2\pi}{(x^2+\pi)}$	0,5	1,9	-3	3	2
11	$y=\ln\pi\sqrt{x^3+x^2}$	-0,6	0,43	-1	1,5	3
12	$y=x^3\cos^2(x+3)$	0,14	0,0027	-2	2	4
13	$y=\sqrt{\frac{1}{2\pi}}e^{-x+1}$	1,5	0,242	0	3	1
14	$y=\sqrt{x}+\sqrt{3-x}$	1	2,4	0	4	2
15	$y=2\,arctgx+\sin\pi x$	1	1,57	-2	2	3
16	$y=x^2-3\sin x$	1,57	-0,53	-0,5	2	4
17	$y=x(1+\cos\pi x)$	0,5	0,5	-1,5	1,5	1
18	$y=\sqrt{x}e^{-x}$	1	0,369	0	3	2
19	$y=tgx-3x^2$	1,2	-1,75	-1,3	1,3	3
20	$y=e^{-x}\sin^2 x$	1,2	0,262	-2	2	4

21	$y=\ln(x-1)\cos^2 x$	3	1,077	1	4	1
22	$y = x\ (1 + \cos\ \ \pi x\ )$	-0,5	-0,5	-1,5	1,5	2
23	$y = \sqrt{e}\left\|x^2-1\right\|-2\right\|$	0,25	1,75	-2	2	3
24	$y = \sqrt{\pi}\, x arctgx$	1	1,4	-2	3	4
25	$y=x-\sin\ \pi x$	0,5	-0,5	-2	3	1
26	$y = x^3 - 5x^2 + 4\sqrt{x}$	0,5	0,875	-1	4	2
27	$y=1+2\cos\pi x-\sin 2x$	0	3	-2	2	3
28	$y=\cos^2 \pi x$ - 2	0	-1	-2	2	4
29	$y = 2x - 5x^{\frac{2}{3}}$	-1	-7	-2	3	1
30	$y = 2(x^3 - 9x^2)^{\frac{1}{3}}$	1	-4	-2	5	2

Таблица 2.5

Вариант формы вывода	Форма вывода информации 78901234567890123456789012345678901234 – позиции
1	Таблица значений I----------I I X I Функция I I----------I I X =… I Y =… I I X =… I Y =… I I----------I Составил: < Ф. И. О. >
2	Таблица

	********** * X =… * Y = … * ********** * X =… * Y =… * ********** Составил: < Ф. И. О. >
3	Таблица значений +----------+ + Аргумент + Функция + +----------+ + X =… + Y =.. + + X =… + Y =… + + ... + ... + +----------+ Составил: < Ф. И. О. >
4	Получено: для заданной функции Y(…) = … для заданной функции Y(…) = … Составил: < Ф. И. О. >