

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное автономное образовательное учреждение
высшего образования
«СЕВЕРО-КАВКАЗСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

Методические указания
по выполнению лабораторных работ
по дисциплине «Объектно-ориентированное программирование»
для студентов направления подготовки
09.03.02 «Информационные системы и технологии»
Профиль «Прикладное программирование в интеллектуальных
информационных системах»

Ставрополь, 2026

ОГЛАВЛЕНИЕ

Лабораторная работа 1. Разработка консольного приложения	3
Лабораторная работа 2. Перенаправление потоков ввода-вывода	18
Лабораторная работа 3. Управление потоком выполнения в программе	31
Лабораторная работа 4. Одномерные массивы	45
Лабораторная работа 5. Многомерные массивы	58
Лабораторная работа 6. Классы. Объектное моделирование	71
Лабораторная работа 7. Конструктор класса. Перегрузка методов класса...	88
Лабораторная работа 8. Проектирование иерархии классов	97
Лабораторная работа 9. Полиморфизм на основе интерфейсов	109
Лабораторная работа 10. Основы работы с файлами	120
Лабораторная работа 11. Обработка данных в файлах	144
Список литературы	165

ЛАБОРАТОРНАЯ РАБОТА 1. РАЗРАБОТКА КОНСОЛЬНОГО ПРИЛОЖЕНИЯ

1.1. Цель и содержание

Цель лабораторной работы: научиться работать с переменными и константами простых типов в C#.

Задачи лабораторной работы:

- научиться объявлять переменные простых типов в языке C#;
- научиться объявлять константы простых типов в языке C#;
- научиться выполнять простейшие действия с переменными и константами.

1.2. Теоретическая часть

1.2.1 Структура консольного приложения

Рассмотрим структуру консольного приложения на языке C#, созданного с использованием средств MS Visual Studio (рис. 1.1).

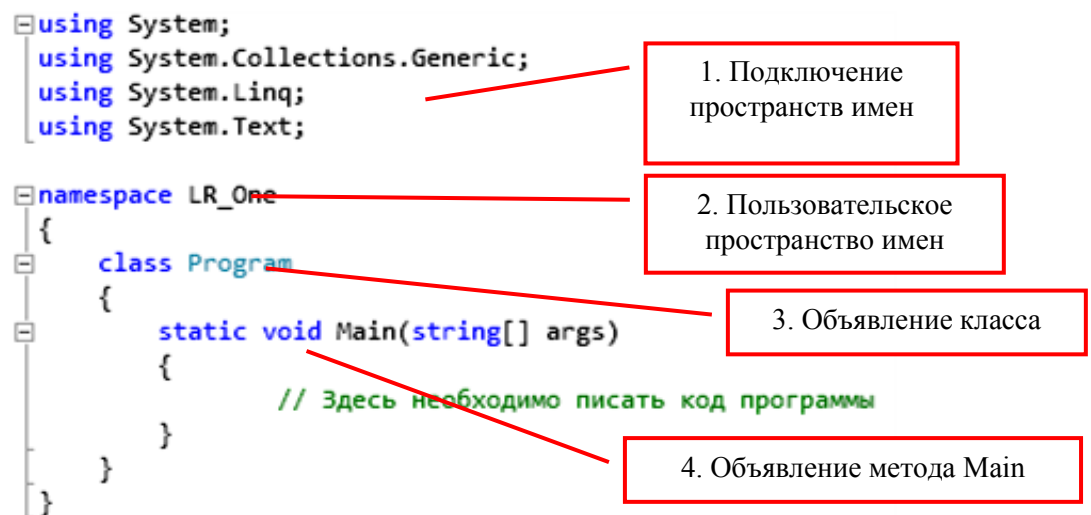


Рисунок 1.1 – Исходный код консольного приложения

В интерактивном редакторе кода среды разработки код может быть свернут/развернут с использованием кнопок +/-, внедренных в код. На рис. 1.1 в области 1

показано подключение пространств имен (библиотек, содержащих стандартные инструменты) с использованием зарезервированного слова `using`.

Программист сам создает свое пространство имен с именем `LR_One` (2), в котором объявляется класс с именем `Program`.

В классе `Program` объявлен один метод – функция `Main` (параметры функции не рассматриваем).

Функция `Main` имеет особенное значение в программировании на языках `C`, `C++` и `C#`.

Функцию `Main` называют «точкой входа», то есть началом выполнения программы. Далее мы рассмотрим приложения, содержащие множество функций. Операционная система знает с какой именно функции начать выполнение программы – с функции `Main`. Очевидно, что имя этой функции менять нельзя. Это должен быть статический метод класса (или структуры), возвращающий либо значение типа `int`, либо `void`. Хотя нередко модификатор `public` указывается явно, поскольку по определению этот метод должен быть вызван извне программы, на самом деле неважно, какой уровень доступа вы назначите методу точки входа. Он запустится, даже если вы пометите его как `private`.

Когда компилируется консольное или Windows-приложение `C#`, по умолчанию компилятор ищет в точности один метод `Main ()` с описанной выше сигнатурой в любом классе и делает его точкой входа программы. Если существует более одного метода `Main ()`, компилятор возвращает сообщение об ошибке.

Обратите внимание на вложенность конструкций на рис. 1.1: в пространство имен `LR_One` вложен класс `Program`, в класс `Program` вложена функция `Main`. В свою очередь, в функции `Main` содержатся инструкции на языке `C#`-код, который начнет выполняться при старте программы.

В `C#`, как и в других `C`-подобных языках, большинство операторов завершаются точкой с запятой (;) и могут продолжаться в нескольких строках без необходимости указания знака переноса. Операторы могут быть объединены в блоки с помощью фигурных скобок (`{ }`). Однострочные комментарии начинаются с двойного слеша (`//`), а многострочные – сослеша со звездочкой (`/*`) и

заканчиваются противоположной комбинацией (*). В этих аспектах язык C# идентичен C++ и Java.

Следует также помнить о том, что язык C# чувствителен к регистру символов. Это значит, что переменные с именами myVar и MyVar являются разными.

Причина присутствия оператора using в файле Program.cs связана с использованием библиотечных классов.

Весь код C# должен содержаться внутри класса. Объявление класса состоит из ключевого слова class, за которым следует имя класса и пара фигурных скобок. Весь код, ассоциированный с этим классом, размещается между этими скобками.

1.2.2 Типы значений C#

Язык C# поддерживает 8 predefined целочисленных типов (таблица 1.1).

Таблица 1.1 – Целочисленные типы C#

Имя типа	Тип CTS	Описание	Диапазон (минимум : максимум)
sbyte	System.SByte	8-битное целое со знаком	-128 : 127
short	System.Int16	16-битное целое со знаком	-32 768 : 32 767
int	System.Int32	32-битное целое со знаком	-2 147 483 648 : 2 147 483 647
long	System.Int64	64-битное целое со знаком	$-2^{63} : 2^{63}-1$
byte	System.Byte	8-битное целое без знака	0 : 255
ushort	System.UInt16	16-битное целое без знака	0 : 65 535
uint	System.UInt32	32-битное целое без знака	0 : $2^{32}-1$
ulong	System.UInt64	64-битное целое без знака	0 : $2^{64}-1$

Язык C# также поддерживает и типы с плавающей точкой (таблица 1.2).

Таблица 1.2 – Типы с плавающей точкой C#

Имя типа	Тип CTS	Описание	Кол-во знаков	Диапазон (минимум : максимум)
float	System.Single	32-битное с плавающей точкой одинарной точности	7	от $\pm 1.5 \times 10^{-45}$ до $\pm 3.4 \times 10^{38}$
double	System.Double	64-битное с плавающей точкой двойной точности	15/16	от $\pm 5.0 \times 10^{-324}$ до $\pm 1.7 \times 10^{308}$

В таблице 1.3 представлен десятичный тип C#. Данный тип реализован для финансовых операций.

Таблица 1.3 – Десятичный тип C#

Имя типа	Тип CTS	Описание	Кол-во знаков	Диапазон (минимум : максимум)
decimal	System.Decimal	128-битное с плавающей точкой в десятичной нотации с высокой точностью	28	от $\pm 1.0 \times 10^{-28}$ до $\pm 7.9 \times 10^{28}$

Как и во многих языках программирования существует булевский тип (таблица 1.4).

Таблица 1.4 – Булевский тип.

Имя типа	Тип CTS	Значения
bool	System.Boolean	true или false

Для хранения одиночных символов в языке C# используется тип char (таблица 1.5)

Таблица 1.5 – Булевский тип.

Имя типа	Тип CTS	Значения
char	System.Char	Представляет отдельный 16-битный (Unicode) символ

Литералы типа char записываются как одиночные, заключенные в одинарные кавычки символы: 'F', 'w', 'ц', 'Я' и т.д.

В переменных типа char можно хранить и специальные символы в виде управляющих последовательностей (таблица 1.6).

Таблица 1.6 – Представление символов в виде управляющих последовательностей.

Управляющая последовательность	Символ
\'	Одиночная кавычка
\"	Двойная кавычка
\\	Обратный слэш
\0	Пусто
\a	Предупреждение (звуковой сигнал)
\b	Забой
\f	Подача формы
\n	Новая строка
\r	Возврат каретки
\t	Символ табуляции
\v	Вертикальная табуляция

Если отдельные символы объединены в строку, то необходимо использовать тип string, который отображается на тип CTS – System.String.

1.3. Методика и порядок выполнения работы

1.3.1. Создание приложения для вывода персональной информации

1. Создайте консольное приложение, для этого выполните следующие действия:

1.1 Выберите команду главного меню *File → New → Project...*

1.2 В открывшемся диалоговом окне (рис. 1.2) выберите необходимые настройки для создаваемого проекта: язык Visual C#; фреймворк: .NET Framework 4 и выше; шаблон: Console Application.

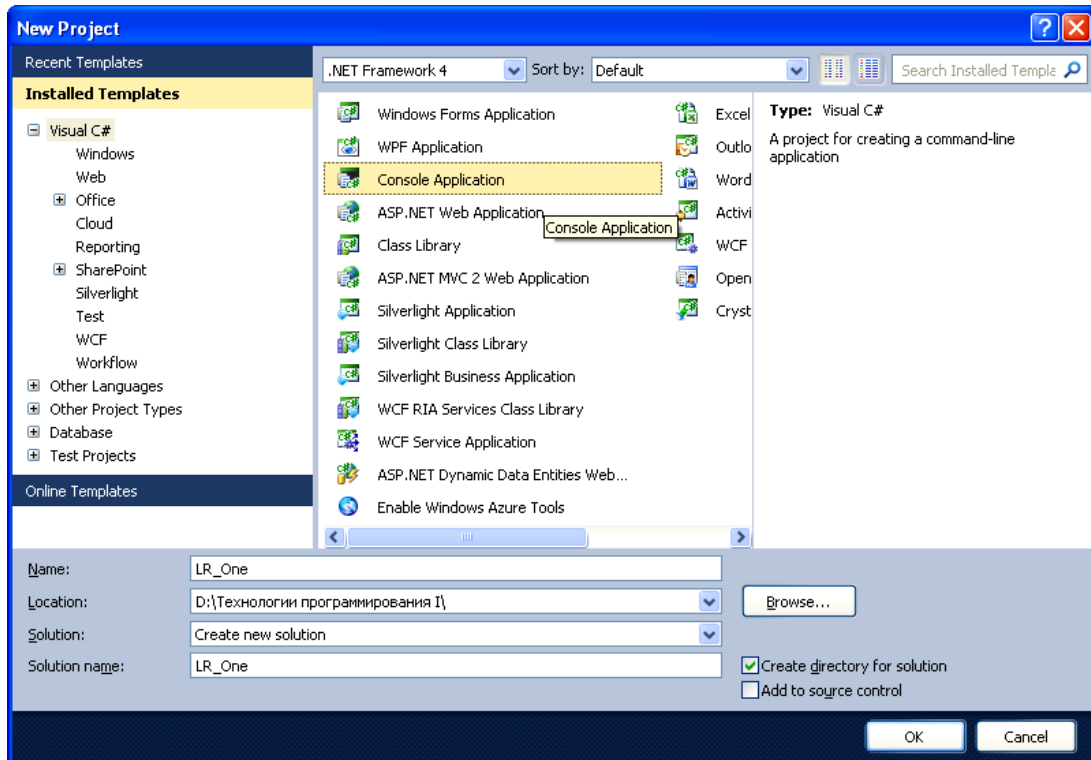


Рисунок 1.2 – Создание нового проекта консольного приложения

1.3 В текстовом поле Name введите имя проекта (например LR_One).

1.4 В текстовом поле Location выберите место сохранения нового проекта.

1.5 Установите флажок-переключатель «Create directory for solution».

1.6 Нажмите кнопку «OK».

2. После выполнения пункта 1 в среде разработки откроется новый созданный проект (рис. 1.3).

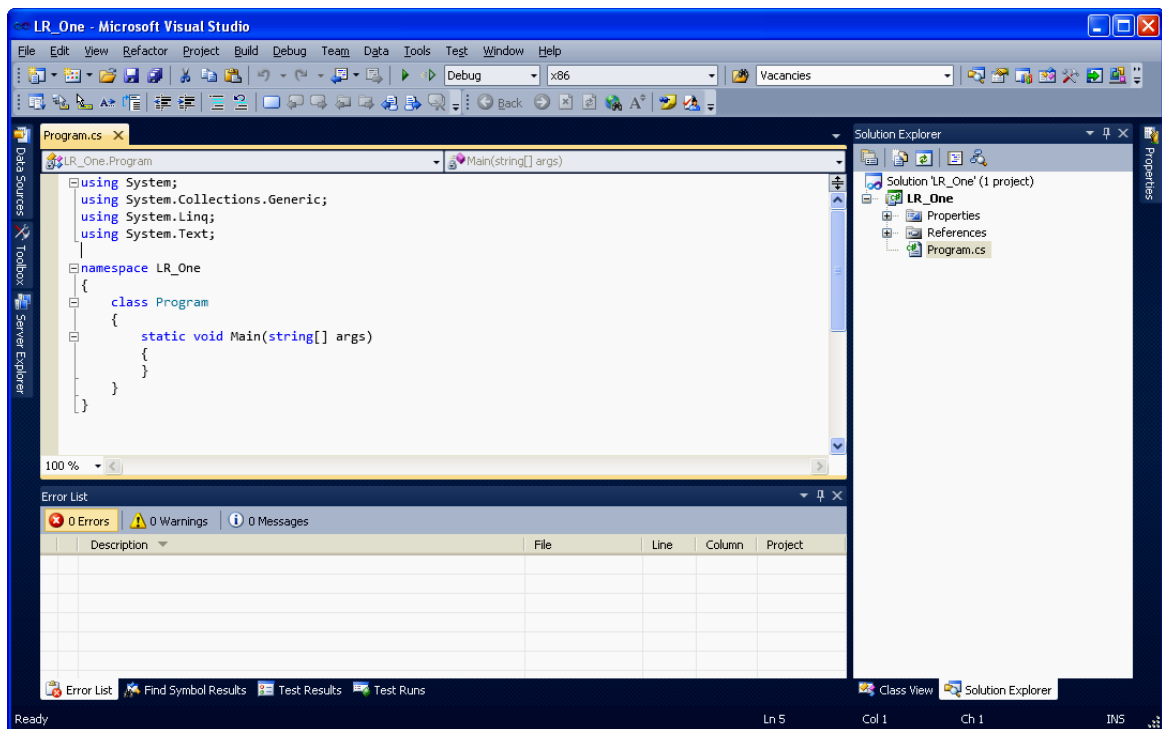


Рисунок 1.3 – Новый проект, загруженный в среду разработки: вкладка «Solution Explorer» отображает состав проекта (нас интересует только файл Program.cs); в левой части окна (сверху) открыт файл Program.cs в редакторе кода

3. На данном этапе необходимо ознакомиться со структурой исходного файла консольного приложения (рис. 1.4).

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;

namespace LR_One
{
    class Program
    {
        static void Main(string[] args)
        {
            // Здесь необходимо писать код программы
        }
    }
}
```

Рисунок 1.4 – Исходный файл консольного приложения

4. Весь код программы необходимо писать внутри функции Main.

5. Для построения сборки (исполняемого exe-файла) выполните команду главного меню *Build → Build Solution* (или использовать горячую клавишу *F6*). После этого сборка создана, но приложение не будет запущено автоматически.

6. Для создания сборки и последующего запуска программы можно воспользоваться командой *Debug → Start Debugging* главного меню среды разработки или нажать кнопку панели инструментов  *Start Debugging (F5)*. Можно также использовать горячую клавишу *F5*.

7. Запустите приложение на выполнение одним из методов, указанных в пункте 6. Окно консольного приложения появится и исчезнет. Это означает, что приложение выполнило все команды, написанные программистом, и завершило свою работу.

8. Для удержания окна на экране измените исходный файл в соответствии с рисунком 1.5. В функции *Main* добавлен вызов только одной команды *Console.ReadKey()* – эта функция останавливает выполнение программы и ждет, когда пользователь нажмет любую клавишу.

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;

namespace LR_One
{
    class Program
    {
        static void Main(string[] args)
        {
            Console.ReadKey();
        }
    }
}
```

Рисунок 1.5 – Исходный файл консольного приложения для предотвращения закрытия окна консольного приложения

9. Запустите измененное приложение, убедитесь, что окно удерживается на экране.

10. Добавьте несколько строк кода в исходный файл (рис. 1.6).

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;

namespace LR_One
{
    class Program
    {
        static void Main(string[] args)
        {
            Console.WriteLine("Лабораторная работа №1");
            Console.WriteLine("");
            Console.WriteLine("Выполнил: Иванов Иван иванович");
            Console.WriteLine("Группа: ИСТБ-121");
            Console.WriteLine("Наименование ЛР: Структура консольного приложения");
            Console.WriteLine("");
            Console.WriteLine("для завершения работы программы нажмите любую клавишу...");

            Console.ReadKey();
        }
    }
}

```

Рисунок 1.6 – Исходный файл консольного приложения для вывода информации на экран.

11. Внимательно изучите исходный код примера на рис. 1.5. Запустите приложение и убедитесь, что отсутствуют ошибки и информация выводится.

1.3.2. Приложение для вычисления значения выражения

1. Создайте новое консольное приложение.
2. Изучите материал в разделе «Теоретическое часть» данной лабораторной работы.
3. Модифицируйте исходный файл как показано на рис. 1.7.

```

1  using System;
2  using System.Collections.Generic;
3  using System.Linq;
4  using System.Text;
5
6  namespace LR_One
7  {
8      class Program
9      {
10         static void Main(string[] args)
11         {
12             int a;
13             int b = 7;
14
15             string str = "Hello, World!!!";
16
17             a = 123;
18
19             Console.ReadKey();
20         }
21     }
22 }

```

Рисунок 1.7 – Объявление переменных в C#

4. Рассмотрим приведенный пример подробнее:

4.1. В строке 12 объявляется переменная типа `int` с именем `a`.

4.2. В строке 13 объявляется переменная типа `int` с именем `b`, причем при объявлении для нее устанавливается начальное значение, равное 7.

4.3. В строке 15 объявляется переменная типа `string` с именем `str` и инициализируется строковым значением «Hello, World!!!».

4.4. В строке 17 переменной `a` присваивается целочисленное значение 123.

5. Запустите приложение на выполнение. У вас должно появиться пустое окно консольного приложения. Очевидно, что исходный код, представленный на рис. 1.7 не предполагает вывода какой-либо информации на экран.

6. Для вывода информации на экран воспользуемся функцией `Console.WriteLine`. Добавим в исходный файл следующие строки (строки 19 – 21 на рис. 1.8).

```

5
6 namespace LR_One
7 {
8     class Program
9     {
10         static void Main(string[] args)
11         {
12             int a;
13             int b = 7;
14
15             string str = "Hello, World!!!";
16
17             a = 123;
18
19             Console.WriteLine(a);
20             Console.WriteLine(b);
21             Console.WriteLine(str);
22
23             Console.ReadKey();
24         }
25     }
26 }
27

```

Рисунок 1.8 – Добавление строк кода для вывода значений объявленных переменных на экран

7. В примере на рис. 1.8 используется простой вывод, то есть имя переменной просто передается в качестве параметра функции Console.WriteLine.

8. Для выполнения форматного вывода (изменения формата представления выводимой информации) необходимо реализовать вывод в следующем виде (рис. 1.9 изменены строки 19-21):

```

1  using System;
2  using System.Collections.Generic;
3  using System.Linq;
4  using System.Text;
5
6  namespace LR_One
7  {
8      class Program
9      {
10         static void Main(string[] args)
11         {
12             int a;
13             int b = 7;
14
15             string str = "Hello, World!!!";
16
17             a = 123;
18
19             Console.WriteLine("Значение переменной a равно {0}", a);
20             Console.WriteLine("a значение переменной b равно {0}", b);
21             Console.WriteLine("значение a+b = {0}+{1} = {2}", a, b, a+b);
22             Console.WriteLine(str);
23
24             Console.ReadKey();
25         }
26     }
27 }

```

Рисунок 1.9 – Вывод информации с использованием форматных строк

9. Внимательно изучите код полученной программы, выполните индивидуальное задание.

1.3.3. Индивидуальное задание

Индивидуальное задание состоит из двух частей:

1. Измените приложение, созданное в ходе выполнения данной лабораторной работы таким образом, чтобы программа выводила на экран следующую информацию (каждый студент должен использовать персональную информацию о себе):

- Название и номер лабораторной работы;
- ФИО студента;
- Группа студента и шифр специальности;
- Дата рождения студента;
- Населенный пункт постоянного места жительства студента;

- Любимый предмет в школе;
- Краткое описание увлечений, хобби, интересов.

2. Создайте второй проект (или добавьте функционал после вывода информации о себе). Объявите требуемые переменные, присвойте им начальные значения (определите самостоятельно, значения какого типа могут принимать переменные), выведите на экран с использованием форматной строки значения переменных и результат вычисления выражения в соответствии с вариантом:

Вариант	Выражение для вычисления
1	$F = a_1 + b - a \cdot (x + y^5)$
2	$Se = w111 + bt - x + y \cdot w$
3	$R_x = a \cdot b + b/t - x + f \cdot i_2$
4	$c = gh + b \cdot q^3 - x + y/w$
5	$H = \frac{g \cdot h}{d17} + b/h1 - \frac{x + y}{4}$
6	$Z = \frac{35}{f} + y \cdot f - \frac{f + y}{4}$
7	$U = \frac{35}{a} \cdot z + z \cdot a - \frac{a + E \cdot t}{4}$
8	$A0 = \frac{35}{G_1} \cdot Zvcw + G_1 \cdot a - \frac{a + Zvcw}{a}$
9	$Se = w111 + bt - x + y \cdot w$
10	$R_x = a \cdot b + b/t - x + f \cdot i_2$
11	$g = w \cdot h + b \cdot q^3 - q^3 + y/w$
12	$ee = \frac{g \cdot h}{d17} + d17/h - \frac{g + d17 + h}{4}$
13	$Zze = \frac{35}{f} + y \cdot f - \frac{f + y}{4}$
14	$t = \frac{35}{a} \cdot z + z \cdot a - \frac{a + Et}{4}$
15	$A0 = \frac{35}{G_1} \cdot Zvcw + G_1 \cdot a - \frac{a + Zvcw}{a}$
16	$Zxy = a_5 + b - a_5 \cdot (b + y)$
17	$ch = \frac{6}{f1} \cdot z + z \cdot f1 - \frac{f1 + Et}{6}$

18	$ch = rx \cdot z + z \cdot f1 - \frac{f1 + Et}{rx}$
19	$_H = k1 + k2 - k3 + \frac{k1 \cdot k2}{k3}$
20	$Fy = a_1 + b - a_1 \cdot (x + y)$
21	$_G_1 = y1 + y2 + y3 - \frac{y1 + y2 + y3}{3}$
22	$R = 2 \cdot x + 2 \cdot y - 4 \cdot x \cdot y + z$
23	$g11 = t \cdot z + z \cdot f1 - \frac{f1 + U}{t}$
24	$Y = Z \cdot (z + x) + z / Z - \frac{z + Z}{x + X}$
25	$U = rx \cdot z + z \cdot Y - \frac{Y + z}{rx}$
26	$c = gh + b \cdot q3 - x + y / w$
27	$H = \frac{g \cdot h}{d17} + b / h1 - \frac{x + y}{4}$
28	$Z = \frac{35}{f} + y \cdot f - \frac{f + y}{4}$
29	$W_1 = rx \cdot z + z \cdot (rx + z) - \frac{Y + z}{rx}$
30	$A0 = \frac{35}{G_1} \cdot Zvcw + G_1 \cdot a - \frac{a + Zvcw}{a}$
31	$Se = w111 + bt - x + y \cdot w$
32	$R_x = 15 + a \cdot b + b / t - x + f \cdot i_2$
33	$g = w \cdot h + b \cdot q3 - q3 + y / w$
34	$ee = \frac{g \cdot h}{d17} + d17 / h - \frac{g + d17 + h}{4}$
35	$Pwd = \frac{12 + f}{f} + y \cdot f - \frac{f + y}{4}$
36	$t = \frac{35}{a} \cdot z + z \cdot a - \frac{a + Et}{4}$
37	$Se = w111 + bt - x + y \cdot w$
38	$R_x = a \cdot b + b / t - x + f \cdot i_2$
39	$t = xx \cdot h + b \cdot q3 - q3 + y / xx$
40	$X = \frac{g \cdot h}{d17} + d17 / h - \frac{g + d17 + h}{4}$

1.4. Контрольные вопросы

1. Какая функция имеет особенное значение при выполнении программы на языках C, C++, C#?
2. Что такое «точка входа» в программе?
3. Как вы понимаете термины «пространства имен», «класс», «метод», «функция»? Напишите определение каждому термину.
4. Что такое переменная? Как объявляется переменная?
5. Как объявляется константа? Чем константа отличается от переменной?
6. Перечислите целочисленные типы C#.
7. Перечислите отличия типов char и string.
8. Какие из перечисленных идентификаторов нельзя использовать в качестве имен пользовательских переменных?
`_1_01`, `b100`, `int`, `double_1`, `_b200`, `MyVar`, `create-var`, `4perem`,
`_5elem`, `zo0`, `wodoo`, `UserCount`, `system_call`, `string`, `System.Double`.
9. Опишите назначение управляющих последовательностей: `'\n'`, `'\t'`, `'\r'`.
10. Имеется список типов: значимые типы, ссылки на функции, ссылочные типы, массивы, делегаты, корреляционные типы, типы по умолчанию. Какие из представленных типов отсутствуют в иерархии типов CTS?
11. К какой ветке дерева типов принадлежат типы `int`, `float`, `double`?
Возможные варианты: встроенные типы по значению, перечисления, самоописываемые типы, упакованные типы по значению.

ЛАБОРАТОРНАЯ РАБОТА 2. ПЕРЕНАПРАВЛЕНИЕ ПОТОКОВ ВВОДА-ВЫВОДА

2.1. Цель и содержание

Цель лабораторной работы: научиться перенаправлять стандартные потоки ввода-вывода.

Задачи лабораторной работы:

- научиться сохранять состояние потоков ввода-вывода;
- научиться объявлять новые потоки, ассоциированные с файлами;
- научиться решать задачи с вводом-выводом в файлы;
- научиться восстанавливать состояние потоков.

2.2. Теоретическая часть

Для выполнения данной лабораторной работы потребуется теоретический материал, изученный в предыдущей лабораторной работе, а также дополнительные сведения о механизмах перенаправления стандартных потоков ввода-вывода.

Работа с файлами будет рассмотрена в данном курсе позже, но можно реализовать чтение и запись в файлы без глубокого знания потоков и других возможностей пространства имен System.IO.

Рассмотрим простое приложение (рисунок 2.1):

```
1  using System;
2
3  namespace Test
4  {
5      class Program
6      {
7          static void Main(string[] args)
8          {
9              int a, b, c;
10             float res = 0;
11             a = Convert.ToInt32(Console.ReadLine());
12             b = Convert.ToInt32(Console.ReadLine());
13             c = Convert.ToInt32(Console.ReadLine());
```

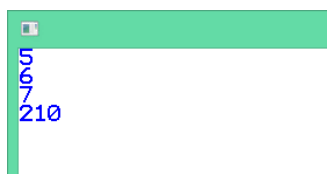
```

14
15         res = a * b * c;
16
17         Console.WriteLine(res);
18
19         Console.ReadKey();
20
21     }
22 }
23

```

Рисунок 2.1 – Пример программы, использующей ввод-вывод

В представленной программе выполняются простые действия: объявляются переменные, считываются с клавиатуры и конвертируются значения, производится вычисление и производится вывод в консоль. На рисунке 2.2 показан результат работы программы.



```

5
6
7
210

```

Рисунок 2.2 – Работа программы, использующей ввод-вывод

Данная программа может быть изменена путем перенаправления стандартных потоков ввода (по умолчанию данный поток связан с клавиатурой) и вывода (по умолчанию связан с экраном, консолью). На рисунке 2.3 показан механизм перенаправления потоков ввода-вывода.

```

1  using System;
2  using System.IO;
3
4  namespace Test
5  {
6      Ссылка: 0
7      class Program
8      {
9          Ссылка: 0
10         static void Main(string[] args)
11         {
12             TextWriter save_out = Console.Out;
13             TextReader save_in = Console.In;
14             var new_out = new StreamWriter(@"output.txt");
15             var new_in = new StreamReader(@"input.txt");
16             Console.SetOut(new_out);
17             Console.SetIn(new_in);

```

```

12     var new_out = new StreamWriter(@"output.txt");
13     var new_in = new StreamReader(@"input.txt");
14     Console.SetOut(new_out);
15     Console.SetIn(new_in);
16
17     int a, b, c;
18     float res = 0;
19     a = Convert.ToInt32(Console.ReadLine());
20     b = Convert.ToInt32(Console.ReadLine());
21     c = Convert.ToInt32(Console.ReadLine());
22
23     res = a * b * c;
24
25     Console.WriteLine(res);
26
27     Console.SetOut(save_out); new_out.Close();
28     Console.SetIn(save_in); new_in.Close();
29 }
30 }
31 }

```

Рисунок 2.3 – Листинг программы, использующей перенаправленный ввод-вывод

Программа, представленная на рисунке 2.3 выглядит аналогично рассмотренной ранее, но добавлены строки: 2, 1-15, 27, 28. То есть сам алгоритм не изменился, дополнительные строки просто перенаправляют ввод-вывод.

В строке 2 подключается пространство имен System.IO, которое содержит необходимые типы для работы с файлами.

В строках 10, 11 сохраняются стандартные потоки System.In и System.Out в переменные save_in и save_out.

В строке 12 создается объект new_out типа StreamWriter, который ассоциируется с файлом output.txt. В строке 13 создается объект new_in типа StreamReader, который ассоциируется с файлом input.txt.

В строках 14 и 15 в качестве новых источников стандартных потоков устанавливаются объекты new_in и new_out. Производится данное действие с использованием методов Console.SetIn() и Console.SetOut().

После таких манипуляций любые чтения из консоли на самом деле производятся из файла input.txt, а любой вывод в консоль производится в файл output.txt.

2.3. Методика и порядок выполнения работы

2.3.1. Учебная задача

Требуется вычислить значения двух выражений *geo* и *my_tmp*:

$$my_tmp = a_71 \times \sqrt{\frac{u2 - test}{x}} \times \frac{zoo}{a71 - zoo},$$
$$geo = \sqrt{\frac{x}{u2}} \times (a_71)^2 \times test,$$

на основании значений переменных *a_71*, *_u2*, *test*, *zoo*, *x*.

Входные данные: вещественные числа *a1*, *a2*, *a3*, *a4*, *a5*, каждое не меньше 0, но не превосходит 10^5 .

Выходные данные: вещественные числа *s* и *k*. Результаты необходимо округлить до сотых. Если вычислить значение выражения невозможно, то вывести слово **ERROR**. Результат округлить до тысячных.

Пример input.txt	Пример output.txt
5 10 20 11 100000	ERROR 50000,000
500,2384 20,0024 14 11 10,1997	8,628 2501697,151

Решение на языке C#. Перед непосредственным программированием решения задачи необходимо проанализировать условие. На основе анализа можно сделать следующие выводы:

1. Переменные *x* и *_u2* не могут быть равными 0.
2. Значение *_u2* - *test* должно быть неотрицательным.
3. Значение *a_71* - *zoo* должно быть отличным от 0.

Решение предложенной задачи представлено на рисунке 2.5

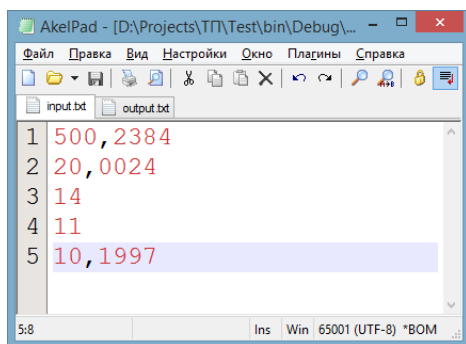
```

1  using System;
2  using System.IO;
3
4  namespace Test
5  {
6      class Program
7      {
8          static void Main(string[] args)
9          {
10             TextWriter save_out = Console.Out;
11             TextReader save_in = Console.In;
12             var new_out = new StreamWriter(@"output.txt");
13             var new_in = new StreamReader(@"input.txt");
14             Console.SetOut(new_out);
15             Console.SetIn(new_in);
16
17             double a_71, _u2, test, zoo, x;
18             double my_tmp, geo;
19             a_71 = Convert.ToDouble(Console.ReadLine());
20             _u2 = Convert.ToDouble(Console.ReadLine());
21             test = Convert.ToDouble(Console.ReadLine());
22             zoo = Convert.ToDouble(Console.ReadLine());
23             x = Convert.ToDouble(Console.ReadLine());
24
25             if ((x == 0) || (a_71 - zoo == 0) || (_u2 - test < 0))
26                 Console.WriteLine("ERROR");
27             else
28             {
29                 my_tmp = a_71 * Math.Sqrt((_u2 - test) / x) * zoo / (a_71 - zoo);
30                 Console.WriteLine(String.Format("{0:0.000}", my_tmp));
31             }
32
33             if (_u2 == 0)
34                 Console.WriteLine("ERROR");
35             else
36             {
37                 geo = Math.Sqrt(x / _u2) * a_71 * a_71 * test;
38                 Console.WriteLine(String.Format("{0:0.000}", geo));
39             }
40
41             Console.SetOut(save_out); new_out.Close();
42             Console.SetIn(save_in); new_in.Close();
43         }
44     }
45 }

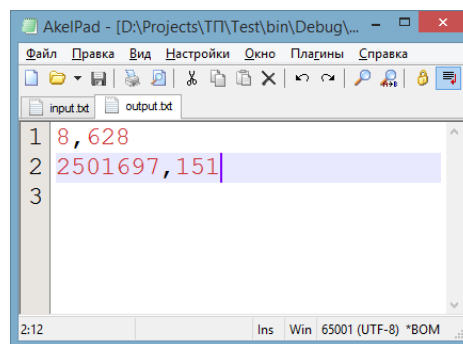
```

Рисунок 2.5 – Решение задачи о вычислении выражения с перенаправлением потоков ввода-вывода

В результате работы программы будет сформирован выходной файл, содержащий результаты вычислений (рисунок 2.6).



а)



б)

Рисунок 2.6 – Файлы, используемые программой в качестве источников ввода-вывода: а) входной файл; б) выходной файл

2.3.2. Индивидуальное задание

Разработайте приложение на языке C# для решения задачи в соответствии с индивидуальным вариантом.

Таблица 2.1 – Индивидуальные задания к лабораторной работе 2.

Вариант	Задание
1	Требуется вычислить площадь треугольника по трем сторонам a, b, c (формула Герона). Заданный треугольник существует. <i>Входные данные:</i> Три вещественных числа a, b, c, каждое из которых больше 1, но не превосходит 1000. <i>Выходные данные:</i> Одно вещественное число – площадь треугольника со сторонами a, b, c. Результат необходимо округлить до тысячных.
2	Требуется вычислить длину окружности радиуса r и площадь образованного ей круга. Число π принять равным 3,14. <i>Входные данные:</i> Одно вещественное число r, $0 < r < 10^5$. <i>Выходные данные:</i> Два вещественных числа: L – длина окружности; S – площадь круга. Результат необходимо округлить до тысячных.
3	Требуется вычислить значение выражения s: $s = a1 \cdot \sqrt{a2 - a3} \cdot \frac{a4}{a5},$ на основании значений переменных a1, a2, a3, a4, a5. <i>Входные данные:</i> Вещественные числа a1, a2, a3, a4, a5, каждое не меньше 0, но не превосходит 10^5. <i>Выходные данные:</i> Одно вещественное число s. Результат необходимо округлить до тысячных. Если вычислить выражение невозможно, программа выводит ERROR.

4	Требуется вычислить значения двух выражений s и k: $s = \frac{a1 \times \sqrt{a2 - a5}}{a3}, k = \sqrt{\frac{a3}{a1}} \times (a2)^2,$ на основании значений переменных a1, a2, a3, a4, a5. <i>Входные данные:</i> вещественные числа a1, a2, a3, a4, a5, каждое не меньше 0, но не превосходит 10^5. <i>Выходные данные:</i> Вещественные числа s и k. Результаты необходимо округлить до сотых. Если вычислить выражение невозможно, программа выводит ERROR.
5	Требуется вычислить значения двух выражений s и k: $s = \frac{\sqrt{a2 - a1 + a3}}{a3} + \frac{a4}{100}, k = \sqrt{\frac{a3 + a4}{a1 - a3}} \times (a2 - a5)^2,$ на основании значений переменных a1, a2, a3, a4, a5. <i>Входные данные:</i> вещественные числа a1, a2, a3, a4, a5, каждое не меньше 0, но не превосходит 10^5. <i>Выходные данные:</i> Вещественные числа s и k. Результаты необходимо округлить до тысячных. Если вычислить выражение невозможно, программа выводит ERROR.
6	Требуется вычислить значения двух выражений s и k: $s = \frac{\sqrt{a2 - a1}}{a3 - a5} + \frac{a1}{a3}, k = \sqrt{\frac{a3 + a4}{3.14 - a3}} \times \frac{1}{(a2 - a5)^2},$ на основании значений переменных a1, a2, a3, a4, a5. <i>Входные данные:</i> вещественные числа a1, a2, a3, a4, a5, каждое не меньше 0, но не превосходит 10^5. <i>Выходные данные:</i> Вещественные числа s и k. Результаты необходимо округлить до десятитысячных. Если вычислить выражение невозможно, программа выводит ERROR.
7	Требуется вычислить значения двух выражений s и k: $s = \frac{\sqrt{a2 - a1}}{\sqrt{a3 - a5}} + \frac{\sqrt{a1}}{a3 - a1}, k = \sqrt{\frac{10000}{a2 - a3}} \times \frac{1}{(a2 - a4)^2},$ на основании значений переменных a1, a2, a3, a4, a5. <i>Входные данные:</i> вещественные числа a1, a2, a3, a4, a5, каждое не меньше 0, но не превосходит 10^5. <i>Выходные данные:</i> Вещественные числа s и k. Результаты необходимо округлить до десятитысячных. Если вычислить выражение невозможно, программа выводит ERROR.

8	Требуется вычислить значения двух выражений s и k: $s = \sqrt{a1 - a2} + \frac{\sqrt{a2 - a3}}{\sqrt{a3 - a4}} + \frac{\sqrt{a1}}{a4 - a5}, k = \sqrt{\frac{1}{a2 - a3}} \times \frac{\sqrt{2}}{(a2 - a4)^2},$ на основании значений переменных a1, a2, a3, a4, a5. <i>Входные данные:</i> вещественные числа a1, a2, a3, a4, a5, каждое не меньше 0, но не превосходит 10^5. <i>Выходные данные:</i> Вещественные числа s и k. Результаты необходимо округлить до тысячных. Если вычислить выражение невозможно, программа выводит ERROR.
9	Требуется вычислить значения двух выражений s и k: $s = \sqrt{a1 - \sqrt{a2 - a3}}, k = \sqrt{\frac{a4 - \sqrt{a5 - a1}}{a2 - a3}},$ на основании значений переменных a1, a2, a3, a4, a5. <i>Входные данные:</i> вещественные числа a1, a2, a3, a4, a5, каждое не меньше 0, но не превосходит 10^5. <i>Выходные данные:</i> Вещественные числа s и k. Результаты необходимо округлить до десятитысячных. Если вычислить выражение невозможно, программа выводит ERROR.
10	Требуется вычислить значения двух выражений s и k: $s = \frac{\sqrt{a - \sqrt{e - b}}}{e}, k = \sqrt{\frac{d + \sqrt{c - 1000}}{ b - a }},$ на основании значений переменных a, b, c, d, e. <i>Входные данные:</i> вещественные числа a, b, c, d, e, каждое не меньше 0, но не превосходит 10^5. <i>Выходные данные:</i> Вещественные числа s и k. Результаты необходимо округлить до сотых. Если вычислить выражение невозможно, программа выводит ERROR.
11	Требуется вычислить значения двух выражений f и g: $f = \frac{1}{\sqrt{apo} - \sqrt{ew}}, g = \sqrt{\frac{boo + \sqrt{ddd - boo}}{css}},$ на основании значений переменных apo, boo, css, ddd, ew. <i>Входные данные:</i> вещественные числа apo, boo, css, ddd, ew, каждое не меньше 0, но не превосходит 10^5. <i>Выходные данные:</i> Вещественные числа f и g. Результаты необходимо округлить до тысячных. Если вычислить выражение невозможно, программа выводит ERROR.

12	Требуется вычислить значения двух выражений <i>f</i> и <i>g</i>: $f = \frac{1}{100} - \frac{1}{apo} - \frac{1}{boo^2}, \quad g = \frac{1}{css^2} + \frac{\sqrt{ew}}{ddd^3},$ на основании значений переменных <i>apo</i>, <i>boo</i>, <i>css</i>, <i>ddd</i>, <i>ew</i>. <i>Входные данные:</i> вещественные числа <i>apo</i>, <i>boo</i>, <i>css</i>, <i>ddd</i>, <i>ew</i>, каждое не меньше 0, но не превосходит 10^5. <i>Выходные данные:</i> Вещественные числа <i>f</i> и <i>g</i>. Результаты необходимо округлить до тысячных. Если вычислить выражение невозможно, программа выводит ERROR.
13	Требуется вычислить значения двух выражений <i>user</i> и <i>pwd</i>: $user = \frac{age}{\sqrt{birth}} + \frac{city}{\sqrt{drive}}, \quad pwd = \sqrt{city - \sqrt{drive - \sqrt{epoch}}},$ на основании значений переменных <i>age</i>, <i>birth</i>, <i>city</i>, <i>drive</i>, <i>epoch</i>. <i>Входные данные:</i> вещественные числа <i>age</i>, <i>birth</i>, <i>city</i>, <i>drive</i>, <i>epoch</i>, каждое не меньше 0, но не превосходит 10^5. <i>Выходные данные:</i> Вещественные числа <i>user</i> и <i>pwd</i>. Результаты необходимо округлить до сотых. Если вычислить выражение невозможно, программа выводит ERROR.
14	Требуется вычислить значения двух выражений <i>f</i> и <i>g</i>: $f = \frac{apo}{boo} + \frac{boo}{css} + \frac{css}{ddd}, \quad g = \sqrt{ddd - \sqrt{ew - apo}},$ на основании значений переменных <i>apo</i>, <i>boo</i>, <i>css</i>, <i>ddd</i>, <i>ew</i>. <i>Входные данные:</i> вещественные числа <i>apo</i>, <i>boo</i>, <i>css</i>, <i>ddd</i>, <i>ew</i>, каждое не меньше 0, но не превосходит 10^5. <i>Выходные данные:</i> Вещественные числа <i>f</i> и <i>g</i>. Результаты необходимо округлить до тысячных. Если вычислить выражение невозможно, программа выводит ERROR.
15	Требуется вычислить значения двух выражений <i>s</i> и <i>k</i>: $s = \frac{\sqrt{a-e}}{\sqrt{b}}, \quad k = \sqrt{\frac{c}{ b-d }},$ на основании значений переменных <i>a</i>, <i>b</i>, <i>c</i>, <i>d</i>, <i>e</i>. <i>Входные данные:</i> вещественные числа <i>a</i>, <i>b</i>, <i>c</i>, <i>d</i>, <i>e</i>, каждое не меньше 0, но не превосходит 10^5. <i>Выходные данные:</i> Вещественные числа <i>s</i> и <i>k</i>. Результаты необходимо округлить до сотых. Если вычислить выражение невозможно, программа выводит ERROR.

16	Требуется вычислить значения двух выражений s и k: $s = \frac{\sqrt{a-e}}{\sqrt{b-a+100}}, \quad k = \sqrt{\frac{c-e}{ b-2d }},$ на основании значений переменных a, b, c, d, e. <i>Входные данные:</i> вещественные числа a, b, c, d, e, каждое не меньше 0, но не превосходит 10^5. <i>Выходные данные:</i> Вещественные числа s и k. Результаты необходимо округлить до сотых. Если вычислить выражение невозможно, программа выводит ERROR.
17	Требуется вычислить значения двух выражений s и k: $s = \frac{\sqrt{b-e+a}}{\sqrt{d-1}}, \quad k = \sqrt{\frac{c-11b}{b-e+a}},$ на основании значений переменных a, b, c, d, e. <i>Входные данные:</i> вещественные числа a, b, c, d, e, каждое не меньше 0, но не превосходит 10^5. <i>Выходные данные:</i> Вещественные числа s и k. Результаты необходимо округлить до сотых. Если вычислить выражение невозможно, программа выводит ERROR.
18	Требуется вычислить значения двух выражений s и k: $s = \frac{1}{\sqrt{a-c}} + \frac{1}{\sqrt{d-8b}}, \quad k = \sqrt{a - \sqrt{2b-c}},$ на основании значений переменных a, b, c, d, e. <i>Входные данные:</i> вещественные числа a, b, c, d, e, каждое не меньше 0, но не превосходит 10^5. <i>Выходные данные:</i> Вещественные числа s и k. Результаты необходимо округлить до тысячных. Если вычислить выражение невозможно, программа выводит ERROR.
19	Требуется вычислить значения двух выражений s и k: $s = \frac{\sqrt{4d-b+1}}{\sqrt{a-2c+1}}, \quad k = \frac{\sqrt{a-\sqrt{b-c}}}{e},$ на основании значений переменных a, b, c, d, e. <i>Входные данные:</i> вещественные числа a, b, c, d, e, каждое не меньше 0, но не превосходит 10^5. <i>Выходные данные:</i> Вещественные числа s и k. Результаты необходимо округлить до десятитысячных. Если вычислить выражение невозможно, программа выводит ERROR.

20	Требуется вычислить значения двух выражений s и k: $s = \frac{b}{\sqrt{c+b-e}+1}, \quad k = \frac{\sqrt{c+\sqrt{b-a}}}{\sqrt{a}},$ на основании значений переменных a, b, c, d, e. <i>Входные данные:</i> вещественные числа a, b, c, d, e, каждое не меньше 0, но не превосходит 10^5. <i>Выходные данные:</i> Вещественные числа s и k. Результаты необходимо округлить до десятитысячных. Если вычислить выражение невозможно, программа выводит ERROR.
21	Требуется вычислить значения двух выражений s и k: $s = \frac{b}{a-\sqrt{c-d}}, \quad k = \frac{100}{\sqrt{c-3d}} + \frac{100}{\sqrt{4e-a}},$ на основании значений переменных a, b, c, d, e. <i>Входные данные:</i> вещественные числа a, b, c, d, e, каждое не меньше 0, но не превосходит 10^5. <i>Выходные данные:</i> Вещественные числа s и k. Результаты необходимо округлить до сотых. Если вычислить выражение невозможно, программа выводит ERROR.
22	Требуется вычислить значения двух выражений s и k: $s = \frac{a^2}{c^2-e^2}, \quad k = \frac{\sqrt{a-c^2}}{\sqrt{c^2-d^3}},$ на основании значений переменных a, b, c, d, e. <i>Входные данные:</i> вещественные числа a, b, c, d, e, каждое не меньше 0, но не превосходит 10^5. <i>Выходные данные:</i> Вещественные числа s и k. Результаты необходимо округлить до тысячных. Если вычислить выражение невозможно, программа выводит ERROR.
23	Требуется вычислить значения двух выражений s и k: $s = \frac{e \cdot a}{c \cdot (b-d)}, \quad k = \frac{\sqrt{(a-b) \cdot (c-d)}}{\sqrt{e-a}},$ на основании значений переменных a, b, c, d, e. <i>Входные данные:</i> вещественные числа a, b, c, d, e, каждое не меньше 0, но не превосходит 10^5. <i>Выходные данные:</i> Вещественные числа s и k. Результаты необходимо округлить до сотых. Если вычислить выражение невозможно, программа выводит ERROR.
24	Требуется вычислить значения двух выражений s и k: $s = \frac{\sqrt{e \cdot a - b \cdot c}}{\sqrt{e-a}}, \quad k = \frac{\sqrt{(a-2b)^3}}{\sqrt{e-a+c}},$ на основании значений переменных a, b, c, d, e. <i>Входные данные:</i> вещественные числа a, b, c, d, e, каждое не меньше 0, но не превосходит 10^5. <i>Выходные данные:</i> Вещественные числа s и k. Результаты необходимо округлить до десятитысячных. Если вычислить выражение невозможно, программа выводит ERROR.

25	Требуется вычислить значения двух выражений s и k: $s = \frac{a}{\sqrt{b-c}} + \frac{b}{\sqrt{15c-d}}, \quad k = \frac{a}{b} + \frac{c}{\sqrt{3d-e}},$ на основании значений переменных a, b, c, d, e. <i>Входные данные:</i> вещественные числа a, b, c, d, e, каждое не меньше 0, но не превосходит 10^5. <i>Выходные данные:</i> Вещественные числа s и k. Результаты необходимо округлить до сотых. Если вычислить выражение невозможно, программа выводит ERROR.
----	---

2.4. Контрольные вопросы

1. Укажите различие между **Console.Write()** и **Console.WriteLine()**.
2. Опишите метод **Console.ReadKey()**. Укажите параметры, возвращаемое значение, применение.
3. Укажите стандартные потоки ввода-вывода в консольном приложении .NET Framework.
4. Укажите различия между методами **Console.ReadLine()** и **Console.Read()**.
5. Программисту необходимо произвести конвертацию значения типа **System.String** в значение типа **float**. Напишите метод, который необходимо использовать.

6. Дан фрагмент кода:

```
double x = 2, y = 2, z = 2, res = 0;
res = Math.Pow(x, y) * z;
```

Укажите значение переменной **res** после выполнения данного фрагмента.

7. Дан фрагмент кода:

```
int a_c = 8;
float b;
b = a_c / 3;
```

Укажите значение переменной **b** после выполнения данного фрагмента.

8. В функции **Main()** содержится следующий код:

```
int work1 = 8, work2 = 17;
int res;
res = work1;
Console.Write(res); ;
```

Поясните, почему в редакторе кода переменная **work2** подчеркнута волнистой линией.

9. Дан листинг Main():

```
static void Main(string[] args)
{
    double p1 = 8, p2 = 14, p3 = 11;
    double res = p3 / (p1 + p2);
    Console.Write(res);
}
```

Какое значение будет выведено в консоли?

10. Дан листинг Main():

```
static void Main(string[] args)
{
    double zoo, geo = 14, hey = 11;
    double res = zoo + geo;
    Console.Write(res + hey);
}
```

Укажите ошибки компиляции (в случае отсутствия ошибок компиляции укажите «ошибок нет»).

ЛАБОРАТОРНАЯ РАБОТА 3. УПРАВЛЕНИЕ ПОТОКОМ ВЫПОЛНЕНИЯ В ПРОГРАММЕ

3.1. Цель и содержание

Цель лабораторной работы: изучить операторы, позволяющие организовывать непоследовательное выполнение программного кода.

Задачи лабораторной работы:

- научиться применять условный оператор if .. else;
- научиться применять операторы цикла for, while, do ... while;
- научиться применять операторы выбора switch.

3.2. Теоретическая часть

3.2.1. Условный оператор

В предыдущих лабораторных работах были рассмотрены вопросы программирования на языке C# с использованием только последовательного выполнения операторов программы.

В языке C# (как и в большинстве языков) можно вводить условия, циклы, ветвления.

Условный оператор в C# позволяет организовать условие типа «если ..., то ..., иначе ...».

Синтаксис условного оператора:

```
if (условие)
    оператор
[else if (условие)
    оператор
...
else
    оператор]
```

В данном синтаксисе *оператор* может быть составным оператором (группа операторов языка, заключенные в фигурные скобки). Условный оператор содержит только одно обязательное зарезервированное слово – `if`. Все остальные конструкции не являются обязательными (в синтаксическом описании на это указывают скобки []).

Пример использования условного оператора:

```
int a = 100, b = 13, c = 23, d = 0, e = 0;

if (a == 100) // Если a равно 100
{
    // Увеличиваем b на 1000
    b += 1000;
}
else if (a < 100) // иначе если a меньше 100
{
    // изменяем значение d
    d = b + c;
    // и изменяем значение e
    e = a + b;
}
else // иначе (остальные варианты не предусмотренные первыми двумя условиями)
    // edtkbxbdfv a на 1
    a++;

// продолжается последовательное выполнение программы
Console.WriteLine("{0}", a);
```

В представленном примере сначала проверяется является ли значение переменной `a` равным 100.

1. Если «Да», то выполняется единственный оператор (увеличение `b` на 1000). На этом выполнение всего условного оператора завершено и программа переходит к выполнению операторов, следующих за ним, то есть к выводу значения `a`.

2. Если первое условие не выполнилось, проверяется второе – является ли `a` меньше 100 (конструкция `else if`). Если это утверждение истинно, то выполняется составной оператор, состоящий из двух – изменение значений переменных `d` и `e`. Затем осуществляется переход к выводу переменной `a`.

3. Если не выполнилось ни одно условие с оператором `if`, то выполняется оператор, указанный после зарезервированного слова `else`. В данном случае это

несоставной оператор (а увеличивается на 1), поэтому фигурные скобки можно не писать.

Во всей этой конструкции только оператор `if` является обязательным. Конструкций `else if` может быть любое количество, а `if` и `else` – только по одному.

Еще один пример условного оператора:

```
int a = 100;

if (a >= 100)
{
    Console.WriteLine("Значение переменной a больше или равно 100");
}
```

В данном случае вывод в консоль выполнится только если `a` больше либо равно 100. Иначе условный оператор ничего не выведет. Фигурные скобки в данном примере можно опустить.

Обратите внимание, что для проверки на равенство используется оператор `==`, а не `=`. Знак «равно» используется в C# для присваивания значений. Следует понимать, что условное выражение, стоящее в конструкции `if` должно возвращать булево значение.

Например, следующий условный оператор всегда будет выполняться:

```
if (true)
{
    Console.WriteLine("Всегда выводится");
}
else
{
    Console.WriteLine("Никогда не выводится");
}
```

3.2.2. Оператор выбора

Синтаксис оператора:

```

switch (перем)
{
    case константа1:
        оператор_1;
        break;
    case константа2:
        оператор_2;
        break;
    ...
    default:
        оператор_n;
        break;
}

```

В данном операторе осуществляется выбор оператора (который может быть составным) в зависимости от значения переменной *перем*. Если *перем* равна значению *константа1*, то выполнится *оператор_1*, если *константа2* – *оператор_2*, и т.д. Если ни одно значение, указанное в каком-либо операторе case, не совпало со значением *перем*, то выполнится оператор, указанный в секции default. Внутри каждой секции case следует указать оператор break, который предотвращает проверку других условий после выполнения данного оператора. Следует обратить внимание, что в секциях case следует использовать только константы (переменные не допускаются).

Пример использование оператора switch ... case:

```
// Организация ответа на действие пользователя:
string text = "1 - Вывод группы \n" +
    "2 - Вывод фамилии преподавателя \n" +
    "3 - Вывод названия предмета \n" +
    "4 - Вывод приветствия \n";

Console.Write(text + "Введите команду > ");
int result = Convert.ToInt32(Console.ReadLine());

switch (result)
{
    case 1: { Console.WriteLine("ИСТБ - 101"); break; }
    case 2: Console.WriteLine("Николаев Евгений Иванович"); break;
    case 3: { Console.WriteLine("Технологии программирования"); break; }
    case 4: Console.WriteLine("Привет !"); break;
    default: Console.WriteLine("Недопустимый вариант !!!"); break;
}
```

Изучите представленный пример самостоятельно.

3.2.3. Цикл for

Циклы позволяют выполнять определенную последовательность операторов до тех пор, пока выполняется некоторое условие. Каждый «круг» выполнения этого блока называется итерацией.

Синтаксис оператора:

```
for (инициализатор; условие; итератор)
    оператор
```

инициализатор – выражение, выполняемое перед первой итерацией цикла.

условие – выражение булевого типа, которое проверяется перед каждой итерацией. Если оно истинно то итерация выполняется, иначе – цикл завершается.

итератор – выражение, вычисляемое после каждой итерации.

Пример применения оператора цикла for (пример 1.1):

```

/*
 * Вычислить сумму чисел от 1 до 1000
 */

// Объявляем переменную, которая хранит значение суммы чисел
System.Int32 Sum = 0;
// Объявляем счетчик цикла
int i;

for (i = 1; i <= 1000; i++)
    Sum += i;

Console.WriteLine("Итог: {0}", Sum);

```

Изучите код самостоятельно (создайте такую программу в VS). Еще один пример для самостоятельного изучения (пример 1.2):

```

/*
 * вычислить сумму всех чисел, делящихся на 3 без остатка
 * и находящихся в диапазоне от 1 до 1000000
 */

long Sum = 0;
int i;

for (i = 1; i <= 1000000; i++)
    if (i % 3 == 0)
        Sum += i;

```

Пример, отображающий возможность применения итератора (пример 1.3):

```

/*
 * вычислить сумму всех чисел, делящихся на 3 без остатка
 * и находящихся в диапазоне от 1 до 1000000
 */

long Sum = 0;
int i;

for (i = 0; i <= 1000000; i += 3)
    Sum += i;

```

Обратите внимание, что примеры 2 и 3 решают одну и ту же задачу.

3.2.4. Цикл while

Цикл while похож на for тем, что является конструкцией с предварительной проверкой условия продолжения цикла. Но синтаксис цикла while более лаконичен:

```
while (условие)
    оператор
```

В данном синтаксисе *оператор* может быть составным оператором (группа операторов языка, заключенные в фигурные скобки).

Следует понимать, что while используется для заранее неизвестного количества повторных выполнений операторов.

Пример выполнения цикла (пример 2.1):

```
/*
 * Цикл будет выполняться пока пользователь не введет 0
 */
bool Stop = false;
int chislo;

while (!Stop)
{
    Console.Write("Введите число > ");
    chislo = Convert.ToInt32(Console.ReadLine());

    if (chislo == 0)
        Stop = true;
}
```

Оператор break уже встречался в конструкции case оператора switch для выхода из case. Но break часто применяется внутри циклов for, while и do ... while. Данный оператор прерывает выполнение цикла и передает управление оператору, следующему за циклом.

Пример 2.2, выполняющий то же самое, что и пример 2.1:

```

/*
 * Цикл будет выполняться пока пользователь не введет 0
 */
bool Stop = true;
int chislo;

while(Stop)
{
    Console.Write("Введите число > ");
    chislo = Convert.ToInt32(Console.ReadLine());

    if (chislo == 0)
    {
        break;
    }
}

```

Следует обратить внимание, что цикл является бесконечным (логическая переменная Stop всегда является true), но выполнение цикла все равно прервется, если пользователь введет 0.

Оператор continue также применяется внутри цикла – он останавливает выполнение текущей итерации и немедленно переходит к выполнению следующей итерации.

3.3. Методика и порядок выполнения работы

3.3.1. Учебная задача

Пример. Требуется написать программу для вычисления значения выражения:

$$Z = \frac{1 \cdot 3 \cdot X^2}{2!} - \frac{3 \cdot 5 \cdot Y^4}{4!} + \frac{5 \cdot 7 \cdot X^6}{6!} - \frac{7 \cdot 9 \cdot Y^8}{8!} + \dots \quad (3.1)$$

Следует реализовать программу таким образом, чтобы все необходимые для вычисления значения выражения исходные значения считывались из входного файла, а результат записывался в выходной файл. Следует учитывать, что необходимо реализовать все три типа цикла при решении задачи, то есть во входном файле необходимо предусмотреть признак, указывающий с использованием какого типа цикла необходимо вычислять выражение.

Решение. Проанализировав выражение (3.1) можно сделать вывод о том, какие именно значения понадобятся, чтобы вычислить выражение для **Z**: нужны значения **X**, **Y**, **N** и **t**. Значение **N** – это количество слагаемых, которые будут складываться; **t** – это тип используемого цикла (например, 0 – цикл for, 1 – цикл while, 2 – цикл do...while). Вид входного файла представлен на рисунке 3.1.

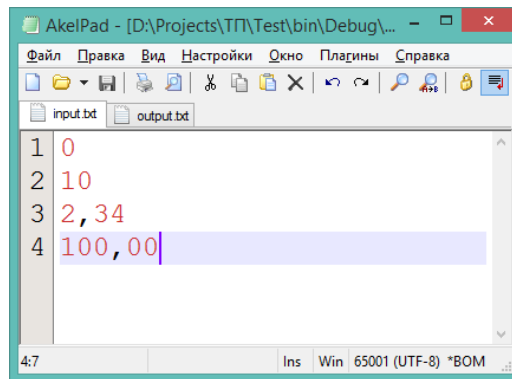


Рисунок 3.1 – Входной файл для решения задачи

Первая строка содержит значение для **t**, вторая – **N**, третья и четвертая – для **X** и **Y** соответственно.

В листинге на рисунке 3.2 представлено решение задачи.

```
1  using System;
2  using System.IO;
3
4  namespace Test
5  {
6      class Program
7      {
8          static void Main(string[] args)
9          {
10             TextWriter save_out = Console.Out;
11             TextReader save_in = Console.In;
12             var new_out = new StreamWriter(@"output.txt");
13             var new_in = new StreamReader(@"input.txt");
14             Console.SetOut(new_out);
15             Console.SetIn(new_in);
16         }
17     }
18 }
```

```

17 int t = 0, N = 1;
18 double X = 0, Y = 0, Z = 0;
19 t = Convert.ToInt32(Console.ReadLine());
20 N = Convert.ToInt32(Console.ReadLine());
21 X = Convert.ToDouble(Console.ReadLine());
22 Y = Convert.ToDouble(Console.ReadLine());
23
24 int i = 1, step = 1;
25 double znam = 1, chisl;
26
27 if (t == 0)
28 {
29     for (i = 1; i <= N; i++)
30     {
31         step = i * 2;
32         znam *= (step - 1) * step;
33         if (i % 2 == 0)
34             chisl = -Math.Pow(Y, step);
35         else
36             chisl = Math.Pow(X, step);
37         Z += (step - 1) * (step + 1) * chisl / znam;
38     }
39 }
40
41 if (t == 1)
42 {
43     i = 1;
44     while (i <= N)
45     {
46         step = i * 2;
47         znam *= (step - 1) * step;
48         if (i % 2 == 0)
49             chisl = -Math.Pow(Y, step);
50         else
51             chisl = Math.Pow(X, step);
52         Z += (step - 1) * (step + 1) * chisl / znam;

```

```

53         i++;
54     }
55 }
56
57 if (t == 2)
58 {
59     i = 1;
60     do
61     {
62         step = i * 2;
63         znam *= (step - 1) * step;
64         if (i % 2 == 0)
65             chisl = -Math.Pow(Y, step);
66         else
67             chisl = Math.Pow(X, step);
68         Z += (step - 1) * (step + 1) * chisl / znam;
69         i++;
70     } while (i <= N);
71 }
72
73 Console.WriteLine(String.Format("{0:0.0000000}", Z));
74
75
76 Console.SetOut(save_out); new_out.Close();
77 Console.SetIn(save_in); new_in.Close();
78 }
79 }
80 }

```

Рисунок 3.2 – Листинг решения учебной задачи

Следует обратить внимание что независимо от выбранного цикла выражение вычисляется и в файл output.txt записывается одно и то же значение.

3.3.2. Индивидуальное задание

Перед выполнением задания требуется самостоятельно определить закономерность изменения членов последовательности, чтобы применить цикл, условный оператор или, если потребуется, оператор выбора. В каждом варианте все входные значения считываются из файла, а результат записывается в файл.

Вариант	Выражение для вычисления
1	$j = -\frac{\sin^3(Y)}{1 \cdot 3} + \frac{\sin^5(X^2)}{3 \cdot 5} - \frac{\sin^7(Y^3)}{5 \cdot 7} + \frac{\sin^9(X^4)}{7 \cdot 9} - \dots$
2	$Z = \frac{1}{1 \cdot 3 \cdot 5} - \frac{X}{2 \cdot 4 \cdot 6} + \frac{Y^2}{3 \cdot 5 \cdot 7} - \frac{X^3}{4 \cdot 6 \cdot 8} + \frac{Y^4}{5 \cdot 7 \cdot 9} - \dots$

3	$Z = \frac{X^2}{1 \cdot 3} - \frac{Y^4}{3 \cdot 5} + \frac{X^6}{5 \cdot 7} - \frac{Y^8}{7 \cdot 9} + \dots$
4	$Z = \frac{Y^2 \cdot X}{2} - \frac{Y^4 \cdot X^3}{4} + \frac{Y^6 \cdot X^5}{6} - \frac{Y^8 \cdot X^7}{8} + \dots$
5	$A = -\frac{\sin(X) \cdot \lg(Y)}{1!} + \frac{\ln(X^3) \cdot \cos(Y^2)}{3!} - \frac{\sin(X^5) \cdot \lg(Y^3)}{5!} + \frac{\ln(X^7) \cdot \cos(Y^4)}{7!} - \dots$
6	$Z = 1 - \frac{X}{2} + \frac{Y^2}{6} - \frac{X^3}{24} + \frac{Y^4}{120} - \dots$ (в знаменателе факториал)
7	$s = -\frac{Y \cdot X^2}{1 \cdot 3} + \frac{X \cdot Y^3}{2 \cdot 4} - \frac{Y \cdot X^4}{3 \cdot 5} + \frac{X \cdot Y^5}{4 \cdot 6} - \dots$
8	$Z = 1 - \frac{\sin(X^2)}{2} + \frac{\cos(Y^3)}{3} - \frac{\sin(X^4)}{4} + \frac{\cos(Y^5)}{5} - \dots$
9	$Z = \frac{1}{1 \cdot 3} - \frac{X}{2 \cdot 4} + \frac{X^2}{3 \cdot 5} - \frac{X^3}{4 \cdot 6} + \dots$
10	$Z = \frac{X^2}{1 \cdot 3} - \frac{Y^4}{3 \cdot 5} + \frac{X^6}{5 \cdot 7} - \frac{Y^8}{7 \cdot 9} + \dots$
11	$p = -\frac{X \cdot \sin(Y)}{1 \cdot 2} + \frac{X^3 \cdot \cos(Y^2)}{3 \cdot 4} - \frac{X^5 \cdot \sin(Y^3)}{5 \cdot 6} + \frac{X^7 \cdot \cos(Y^4)}{7 \cdot 8} - \dots$
12	$p = -\frac{X}{1 \cdot 2 \cdot 3} + \frac{Y^3}{3 \cdot 4 \cdot 5} - \frac{X^5}{5 \cdot 6 \cdot 7} + \frac{Y^7}{7 \cdot 8 \cdot 9} - \dots$
13	$Z = 1 - \frac{\sin(X^2) + Y}{2} + \frac{\cos(Y^3) + X}{3} - \frac{\sin(X^4) + Y}{4} + \frac{\cos(Y^5) + X}{5} - \dots$
14	$Z = \frac{Y + X}{1 \cdot 2} - \frac{Y^2 - X^2}{2 \cdot 4} + \frac{X^3 + Y^3}{4 \cdot 6} - \frac{X^4 - Y^4}{6 \cdot 8} + \dots$
15	$p = 1 - \frac{\cos(X) \cdot \sin^2(Y)}{1 \cdot 2} + \frac{\cos^4(X) \cdot \sin^3(Y)}{3 \cdot 4} - \frac{\cos^5(X) \cdot \sin^6(Y)}{5 \cdot 6} + \dots$
16	$s = -\frac{\sin(Y) \cdot X^2}{1 \cdot 3} + \frac{X \cdot \cos(Y^3)}{2 \cdot 4} - \frac{\sin(Y) \cdot X^4}{3 \cdot 5} + \frac{X \cdot \cos(Y^5)}{4 \cdot 6} - \dots$
17	$Z = -\frac{\operatorname{tg}(Y) \cdot \operatorname{ctg}(X^2)}{1 \cdot 3} + \frac{\operatorname{tg}(X) \cdot \operatorname{ctg}(Y^3)}{2 \cdot 4} - \frac{\operatorname{tg}(Y) \cdot \operatorname{ctg}(X^4)}{3 \cdot 5} + \frac{\operatorname{tg}(X) \cdot \operatorname{ctg}(Y^5)}{4 \cdot 6} - \dots$
18	$Z = \frac{1}{1!} - \frac{X}{2!} + \frac{Y^2}{3!} - \frac{X^3}{4!} + \frac{Y^4}{5!} - \dots$
19	$Z = \frac{\cos(X) + Y }{1!} - \frac{\cos^2(Y) + X }{2!} + \frac{\cos^3(X) + Y }{3!} - \frac{\cos^4(Y) + X }{4!} + \dots$
20	$Z = \frac{Y^2 \cdot X}{2+1} - \frac{Y^2 \cdot X^4}{4-2} + \frac{X^4 \cdot Y^6}{6+4} - \frac{X^8 \cdot Y^6}{8-6} + \dots$

3.4. Контрольные вопросы

1. Опишите, каким образом оформляется комментарий в языке C#.

2. Какое ключевое (зарезервированное) слово в условном операторе является обязательным?

3. Дан фрагмент кода:

```
double x = 2, y = 3, z = 4, res;  
Boolean yes = true;  
if (!yes)  
{  
    res = Math.Pow(x, y) * z;  
}  
else  
{  
    res = Math.Pow(z, x) * y;  
}
```

Чему равно значение переменной **res** после выполнения данного фрагмента?

4. Дан фрагмент кода:

```
int counter = 10;  
const int N = 10;  
for (int i = 0; i < N; i++)  
{  
    counter += i;  
}
```

Чему равно значение переменной **counter** после выполнения данного фрагмента?

5. Дан фрагмент кода:

```
int seo = 0;  
const int N = 5;  
for (int i = 1; i <= N; i++)  
{  
    seo += 2;  
}
```

Чему равно значение переменной **seo** после выполнения данного фрагмента?

6. Что будет выведено в консоль в результате выполнения следующей программы?

```

static void Main(string[] args)
{
    int a = 0, b = 5, c = 0;
    for (;a<b;)
    {
        a+=5;
        b+=2;
        c++;
    }
    Console.Write(c);
    Console.ReadKey();
}

```

7. Что будет выведено в консоль в результате выполнения следующей программы?

```

static void Main(string[] args)
{
    int a = 0, b = 5, c = 0;
    while (a<b)
    {
        a+=5;
        b+=2;
        c++;
    }
    Console.Write(c);
    Console.ReadKey();
}

```

8. Что будет выведено в консоль в результате выполнения следующей программы?

```

static void Main(string[] args)
{
    int a = 3, z = 0;
    Boolean _stop = false;
    while (!_stop)
    {
        z+=a;
        if (z > 100)
            _stop = true;
    }
    Console.Write(z);
    Console.ReadKey();
}

```

ЛАБОРАТОРНАЯ РАБОТА 4. ОДНОМЕРНЫЕ МАССИВЫ

4.1. Цель и содержание

Цель лабораторной работы: изучить типы и принципы работы с одномерными массивами.

Задачи лабораторной работы:

- научиться инициализировать одномерные массивы;
- научиться реализовывать простейшие алгоритмы на одномерных массивах;
- научиться вычислять простейшие агрегированные показатели на одномерных массивах.

4.2. Теоретическое обоснование

Для работы с набором значений одного типа используются массивы.

Массив – это структура данных, содержащая множество элементов одного и того же типа.

Массив определяется типом элементов, содержащимся в нем, за которым следуют квадратные скобки и имя переменной.

Примеры объявления массивов:

```
// Объявление массива целочисленных элементов
int[] mas_int;

// Объявление массива элементов типа double
double[] mas_d;

// Объявление массива элементов типа string
string[] MassivStr;
```

При объявлении массива не указывается его размер. Для выделения памяти для элементов массива необходимо использовать следующий синтаксис (для массивов, объявленных в предыдущем примере):

```
// Спецификация размера массивов
mas_d = new double[5];
mas_int = new Int32[100];
MassivStr = new string[100];
```

После данной инициализации можно обращаться к элементам массивов.

После такой инициализации нельзя изменять размер массива.

Объявить и инициализировать массив можно в одной строке:

```
float[] keof = new float[1000];
```

При инициализации можно также присвоить значения элементам массива (все три способа присвоения элементов массива эквивалентны):

```
int[] m1 = new int[5] {1, 23, 2, 100, 7};  
int[] m2 = new int[] {23, 12, 100, 101, 0};  
int[] m3 = {0, 1, 2, 56, 90};
```

Там, где программист не указал явно размер массива, компилятор определит его самостоятельно. Очевидно, что задать таким образом большое количество элементов неудобно. Для доступа к элементам массивов используется индексатор.

После того как массив объявлен и инициализирован, можно производить доступ к элементам массива для чтения и записи через индексатор. Индексатор – это целочисленный номер элемента массива.

Индексатор начинается с 0 (для первого элемента массива) и заканчивается числом равным количеству элементов в массиве минус 1.

Например, определим массив и попробуем обратиться к его элементам:

```
// Объявление и инициализация массива  
int[] mas = new int[4];  
  
// Установка значений элементов массива  
mas[0] = 1000; // для первого элемента  
mas[1] = 45; // для второго элемента  
  
// Считывание значений  
mas[2] = mas[0]; // считываем значение первого и присваиваем его третьему элементу  
int val = mas[1];  
int a = mas[3]; // возникнет ОШИБКА! Т.к. четвертый элемент не инициализирован  
int Oo = mas[4]; // возникнет ОШИБКА! Т.к. пятого элемента в массиве нет
```

Часто (когда размер массива большой) используют циклы для обращения к элементам массива:

```

/*
 * В цикле последовательно запрашиваются у пользователя
 * значения для инициализации элементов массива
 */
for (int i = 0; i < massiv.Length; i++)
{
    Console.Write("Введите значение {0}-го элемента > ", i+1);
    massiv[i] = Convert.ToInt32(Console.ReadLine());
}

```

Для больших массивов есть смысл сгенерировать значения элементов массива случайным образом (используя генератор случайных чисел):

```

int[] Pro = new int[1000];

// Объявление переменной класса Random
Random rnd = new Random(1);

int i = 0;

while (i < Pro.Length)
{
    // Вызов метода Next класса Random
    Pro[i] = rnd.Next();
    i++;
}

// Вывод элементов массива в обратном порядке
for (i = Pro.Length-1; i >= 0; i--)
    Console.WriteLine("Элемент {0} равен {1}", i, Pro[i]);

```

Внимательно изучите представленный код.

4.3. Методика и порядок выполнения работы

4.3.1. Учебная задача

Условие. Дана последовательность N целых чисел. Вычислить среднее арифметическое этих чисел. Найти среди исходных N чисел все числа, которые не меньше среднего арифметического. Для выполнения задачи необходимо написать программу-генератор исходных массивов.

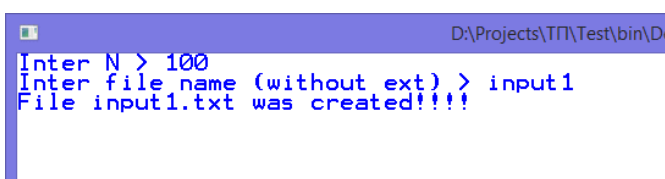
Решение. Перед непосредственно кодированием задачи необходимо проанализировать условие и разбить задачу на подзадачи. Особое внимание нужно уделить написанию программы-генератора исходных данных, то есть сначала

необходимо написать программу, которая позволит автоматически создавать файлы, которые будут содержать исходные данные для решения учебной задачи. Код программы-генератора массивов представлен на рисунке 4.1.

```
1  using System;
2  using System.IO;
3
4  namespace Test
5  {
6      class Program
7      {
8          static void Main(string[] args)
9          {
10             int N;
11             String FileName;
12
13             Console.Write("Enter N > ");
14             N = Convert.ToInt32(Console.ReadLine());
15             Console.Write("Enter file name (without ext) > ");
16             FileName = Console.ReadLine();
17             FileName += ".txt";
18
19             TextWriter save_out = Console.Out;
20             var new_out = new StreamWriter(FileName);
21             Console.SetOut(new_out);
22
23             Console.WriteLine(N);
24
25             Random r = new Random(DateTime.Now.Millisecond);
26             int x = 0;
27             for (int i = 0; i < N; i++)
28             {
29                 x = r.Next(1000);
30                 Console.Write(x + " ");
31             }
32
33             Console.SetOut(save_out); new_out.Close();
34             Console.WriteLine("File " + FileName + " was created!!!!");
35             Console.ReadKey();
36         }
37     }
38 }
```

Рисунок 4.1 – Листинг программы для генерации исходных файлов

Вывод данной программы в консоли представлен на рисунке 4.2.



The screenshot shows a console window with a blue title bar. The text inside the window is as follows:

```
Inter N > 100
Inter file_name (without ext) > input1
File input1.txt was created!!!!
```

Рисунок 4.2 – Вывод в консоль программы для генерации исходных файлов

С использованием данного генератора можно сформировать следующие файлы (рисунок 4.3), содержащие массивы данных для обработки и решения задач по тематике одномерных массивов.

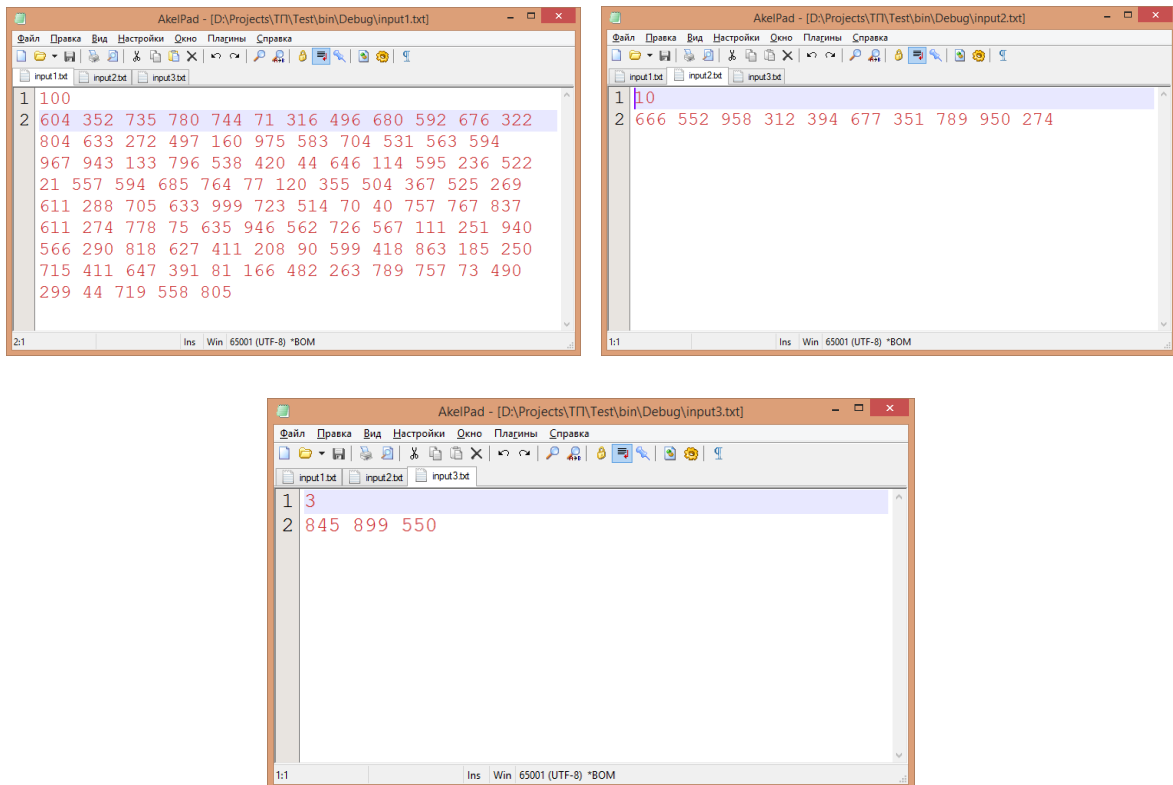


Рисунок 4.3 – Файлы, сгенерированные программой-генератором

Понятно, что для решения учебной задачи можно сформировать исходный файл вручную, но для больших массивов это будет сделать проблематично.

Теперь напишем программу, которая на основании данных, содержащихся во входном файле, вычисляет среднее арифметическое всех чисел и выводит все исходные числа, которые не меньше среднего арифметического (рисунок 4.4).

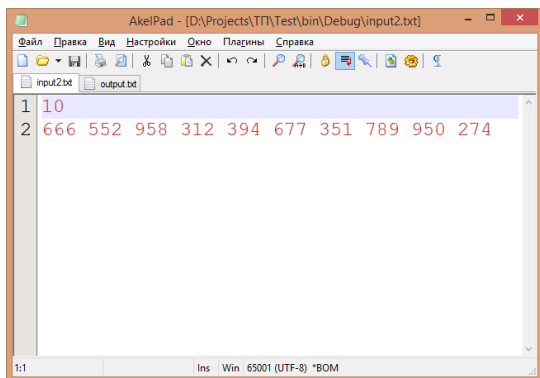
```

1  using System;
2  using System.IO;
3
4  namespace Test
5  {
6      class Program
7      {
8          static void Main(string[] args)
9          {
10             TextWriter save_out = Console.Out;
11             TextReader save_in = Console.In;
12             var new_out = new StreamWriter(@"output.txt");
13             var new_in = new StreamReader(@"input2.txt");
14             Console.SetOut(new_out);
15             Console.SetIn(new_in);
16
17             // Чтение данных из файла и инициализация массива
18             int N = Convert.ToInt32(Console.ReadLine());
19             String str_all = Console.ReadLine();
20             string[] str_elem = str_all.Split(' ');
21
22             int[] mas = new int[N];
23             for (int i = 0; i < N; i++)
24             {
25                 mas[i] = Convert.ToInt32(str_elem[i]);
26             }
27
28             // Вычисление среднего арифметического
29             int s = 0;
30             float sa = 0;
31             for (int i = 0; i < N; i++)
32             {
33                 s += mas[i];
34             }
35             sa = 1.0f * s / N;
36
37             // Вывод среднего арифметического и требуемых значений
38             Console.WriteLine(string.Format("{0:0.000000}", sa));
39             for (int i = 0; i < N; i++)
40             {
41                 if (mas[i] >= sa)
42                     Console.Write(mas[i] + " ");
43             }
44
45             Console.SetOut(save_out); new_out.Close();
46             Console.SetIn(save_in); new_in.Close();
47         }
48     }
49 }

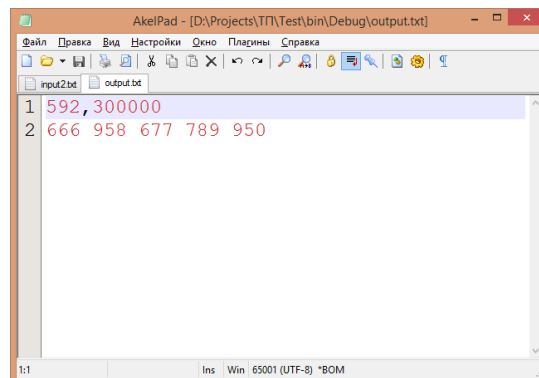
```

Рисунок 4.4 – Вычисление среднего арифметического и вывод требуемых чисел

В результате работы программы для файла input2.txt будет сформирован следующий output.txt (рисунок 4.5):



а)



б)

Рисунок 4.5 – Файлы учебной задачи: а) входной; б) выходной

4.3.2. Индивидуальное задание

Для инициализации исходной матрицы необходимо использовать программу-генератор, разработанную в ходе выполнения учебной задачи. Целесообразно использовать ограничение на генерируемые числа, например, ограничить значения случайно генерируемых чисел величиной 100, 10 и т.д., чтобы вывод программы был более читабельным. Алгоритм решения должен работать для массивов любой величины, поэтому необходимо протестировать программу на различных входных данных.

Вариант	Задание
1	Во входном файле содержится две строки: первая содержит одно целое число N (количество чисел во второй строке), вторая строка содержит N вещественных чисел (каждое от 0 до 10^5). В результате работы программы должен быть сформирован выходной файл, который содержит следующие значения: 1) количество чисел, больших среднего арифметического исходных чисел; 2) сумма всех чисел, меньших среднего арифметического; 3) максимальное число.

2	Во входном файле содержится две строки: первая содержит одно целое число N (количество чисел во второй строке), вторая строка содержит N вещественных чисел (каждое от 0 до 10^5). В результате работы программы должен быть сформирован выходной файл, который содержит следующие значения: 1) произведение чисел, меньших 50 (если таких чисел нет, то выводится 0); 2) сумма всех чисел, больших среднего арифметического; 3) минимальное число.
3	Во входном файле содержится две строки: первая содержит одно целое число N (количество чисел во второй строке), вторая строка содержит N вещественных чисел (каждое от 0 до 10^5). В результате работы программы должен быть сформирован выходной файл, который содержит следующие значения: 1) произведение чисел, меньших 50 (если таких чисел нет, то выводится 0); 2) все числа, большие среднего арифметического через пробел.
4	Во входном файле содержится две строки: первая содержит одно целое число N (количество чисел во второй строке), вторая строка содержит N вещественных чисел (каждое от 0 до 10^5). В результате работы программы должен быть сформирован выходной файл, который содержит следующие значения: 1) среднее арифметическое максимального и минимального элементов; 2) среднее арифметическое всех элементов; 3) все числа, которые больше среднего арифметического максимального и минимального элементов.
5	Во входном файле содержится две строки: первая содержит одно целое число N (количество чисел во второй строке), вторая строка содержит N вещественных чисел (каждое от 0 до 10^5). В результате работы программы должен быть сформирован выходной файл, который содержит следующие значения: 1) максимальный элемент; 2) среднее арифметическое всех элементов; 3) модифицированный массив, в котором все элементы исходного, меньшие среднего арифметического, заменены на 0.
6	Во входном файле содержится две строки: первая содержит одно целое число N (количество чисел во второй строке), вторая строка содержит N вещественных чисел (каждое от -10^5 до 10^5). В результате работы программы должен быть сформирован выходной файл, который содержит следующие значения: 1) максимальный отрицательный элемент; 2) минимальный положительный элемент; 3) список элементов, не превышающих по модулю 50.

7	Во входном файле содержится две строки: первая содержит одно целое число N (количество чисел во второй строке), вторая строка содержит N целых чисел (каждое от -10^5 до 10^5). В результате работы программы должен быть сформирован выходной файл, который содержит следующие значения: 1) максимальный четный элемент; 2) минимальный нечетный элемент; 3) сумму элементов, которые делятся на 5.
8	Во входном файле содержится две строки: первая содержит одно целое число N (количество чисел во второй строке), вторая строка содержит N вещественных чисел (каждое от -10^5 до 10^5). В результате работы программы должен быть сформирован выходной файл, который содержит следующие значения: 1) среднее арифметическое всех элементов; 2) максимальный элемент; 3) модифицированный массив, в котором все элементы не превышающие по модулю 1000 заменены на 0.
9	Во входном файле содержится две строки: первая содержит одно целое число N (количество чисел во второй строке), вторая строка содержит N целых чисел (каждое от -10^5 до 10^5). В результате работы программы должен быть сформирован выходной файл, который содержит следующие значения: 1) максимальный элемент среди всех кратных 5; 2) минимальный элемент среди всех кратных 3; 3) сумму элементов, которые делятся на 10.
10	Во входном файле содержится две строки: первая содержит одно целое число N (количество чисел во второй строке), вторая строка содержит N вещественных чисел (каждое от -10^5 до 10^5). В результате работы программы должен быть сформирован выходной файл, который содержит следующие значения: 1) второй по величине элемент; 2) минимальный положительный элемент; 3) список элементов, не превышающих по модулю 1000.
11	Во входном файле содержится две строки: первая содержит одно целое число N (количество чисел во второй строке), вторая строка содержит N целых чисел (каждое от -10^5 до 10^5). В результате работы программы должен быть сформирован выходной файл, который содержит следующие значения: 1) сумму элементов, кратных 7; 2) произведение элементов, кратных 5; 3) список элементов кратных 10.

12	Во входном файле содержится две строки: первая содержит одно целое число N (количество чисел во второй строке), вторая строка содержит N вещественных чисел (каждое от -10^5 до 10^5). В результате работы программы должен быть сформирован выходной файл, который содержит следующие значения: 1) максимальный элемент, не превышающий 1000; 2) минимальный по модулю элемент; 3) список элементов, больших 5000 и меньших 7000.
13	Во входном файле содержится две строки: первая содержит одно целое число N (количество чисел во второй строке), вторая строка содержит N целых чисел (каждое от -10^5 до 10^5). В результате работы программы должен быть сформирован выходной файл, который содержит следующие значения: 1) количество элементов, кратных 5; 2) максимальный элемент; 3) список элементов, кратных 8.
14	Во входном файле содержится две строки: первая содержит одно целое число N (количество чисел во второй строке), вторая строка содержит N вещественных чисел (каждое от -10^5 до 10^5). В результате работы программы должен быть сформирован выходной файл, который содержит следующие значения: 1) сумму элементов, не превышающих по модулю 1000; 2) разность максимального и минимального элемента; 3) список отрицательных элементов.
15	Во входном файле содержится две строки: первая содержит одно целое число N (количество чисел во второй строке), вторая строка содержит N целых чисел (каждое от -10^5 до 10^5). В результате работы программы должен быть сформирован выходной файл, который содержит следующие значения: 1) максимальный четный элемент; 2) минимальный нечетный элемент; 3) модифицированный массив, в котором все элементы, меньшие среднего арифметического заменены на +.

4.4. Контрольные вопросы

1. Дайте определение массива в языке C#.
2. Укажите строки, в которых верно выполнено объявление массива:

```
int[] m;
```

```
System.String[] names = { "Piro", "Nadi", "Deny"};
```

```
float[,] points;
```

```
System.Int64[] big;
```

```
char[] c = { 'z', 'y', 'x' };
```

```
int mas[];
```

```
float[3] points = { 1.0, 2.0, 3.0 };
```

```
double coord;
```

```
string[] str = new string[10];
```

3. Дан фрагмент кода:

```
int[] z = new int[150];
for (int i = 0; i < z.Length; i++)
{
    z[i] = i * 2;
}
int res = z[8];
```

Какое значение содержится в переменной **res** после выполнения фрагмента?

4. Что будет выведено в консоль в результате работы программы?

```
static void Main(string[] args)
{
    Random r = new Random();
    int[] z = new int[150];
    for (int i = 0; i < z.Length; i++)
    {
        z[i] = r.Next();
    }
    int res = 0;
    for (int i = 0; i < z.Length; i++)
    {
        res += z[i];
    }

    Console.Write(res);
    Console.ReadKey();
}
```

5. Поясните, как связаны величины: количество элементов в массиве, максимальное значение индекса массива, минимальное значение индекса массива.

6. Что будет выведено в консоль в результате работы программы?

```
static void Main(string[] args)
{
    double[] z = { 3.0, 2.5, 10.5, 11.5, 3.01 };

    for (int i = 1; i < z.Length; i++)
    {
        z[i]++;
    }

    Console.Write(z[0]);
    Console.ReadKey();
}
```

7. Какое значение будет выведено в результате выполнения программы?

```
static void Main(string[] args)
{
    float[] z = new float[5];
    float s = 0;
    for (int i = 0; i <= z.Length; i++)
    {
        s += z[i];
    }

    Console.Write(s);
    Console.ReadKey();
}
```

8. Что будет выведено в результате выполнения программы?

```
static void Main(string[] args)
{
    string[] имена = new string[5];
    имена[0] = "Иван";
    имена[1] = "Марк";
    имена[2] = "Denis";
    имена[3] = "Sergey";
    имена[4] = "Алина";
    имена[5] = "Павел";

    Console.Write(имена);
    Console.ReadKey();
}
```

9. Что будет выведено в результате выполнения программы?

```
static void Main(string[] args)
{
    string[] имена = new string[5];
    int[] номера = { 0, 1, 2, 3, 4 };
    имена[0] = "Иван";
    имена[1] = "Марк";
    имена[2] = "Denis";
    имена[3] = "Sergey";
    имена[4] = "Алина";

    string best_name = "";
    for (int i = 0; i < номера.Length; i++)
    {
        best_name = имена[номера[i]];
    }

    Console.Write(best_name);
    Console.ReadKey();
}
```

ЛАБОРАТОРНАЯ РАБОТА 5. МНОГОМЕРНЫЕ МАССИВЫ

5.1. Цель и содержание

Цель лабораторной работы: изучить типы и принципы работы с многомерными массивами.

Задачи лабораторной работы:

- научиться инициализировать двумерные массивы;
- научиться реализовывать простейшие алгоритмы на многомерных массивах;
- научиться вычислять простейшие агрегированные показатели на многомерных массивах.

5.2. Теоретическое обоснование

Обычные массивы (одномерные) индексируются одним числом. Многомерные массивы индексируются несколькими индексаторами. Очевидно, что двумерный массив можно отождествлять с матрицей, трехмерный – с кубом.

Принципы объявления, инициализации и обращения к элементам остаются теми же, что и для одномерных массивов:

```
// Объявляем и инициализируем двумерный массив
// - матрицу целых размером 3x3
int[,] matrix = new int[3, 3];

// Присваиваем значение диагональным элементам
matrix[0, 0] = 1;
matrix[1, 1] = 120;
matrix[2, 2] = 8;

// Объявляем матрицу с одновременным присвоением
// значений элементов
int[,] mm = {
    {0, 23, 100},
    {6, 12, 23},
    {0, 0, 5}
};

// Объявляем трехмерный массив
double[, ,] org = new double[100, 100, 50];

// Присваиваем значение 34 элементу с индексами (0, 0, 0)
org[0, 0, 0] = 34;
```

Как и в случае одномерных массивов, для работы с многомерными применяются циклы.

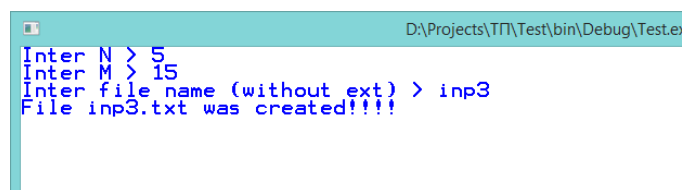
5.3. Методика и порядок выполнения работы

5.3.1. Учебная задача

Условие. Дана исходная матрица размером $M \times N$ из целых чисел, принадлежащих отрезку $[-10^3, 10^3]$. В выходной файл необходимо вывести следующие значения:

- 1) исходную матрицу;
- 2) максимальный элемент для каждой строки;
- 3) среднее арифметическое матрицы и модифицированную матрицу, в которой все элементы, большие среднего арифметического элементов, заменены на $+$, остальные на $-$.

Решение. По аналогии с одномерными массивами перед выполнением основного задания необходимо написать программу, которая будет генерировать входные файлы для задачи. Данная программа-генератор должна генерировать матрицы различного размера и типа (данный пункт реализуйте самостоятельно по аналогии с рассмотренной реализацией генератора). На рисунке 5.1 представлено окно программы-генератора, а на рисунке 5.2 представлены примеры входных файлов, полученных при использовании разработанного генератора.



```
D:\Projects\ТП\Тест\bin\Debug\Test.e...
Enter N > 5
Enter M > 15
Enter file name (without ext) > inp3
File inp3.txt was created!!!!
```

Рисунок 5.1 – Возможный интерфейс программы-генератора

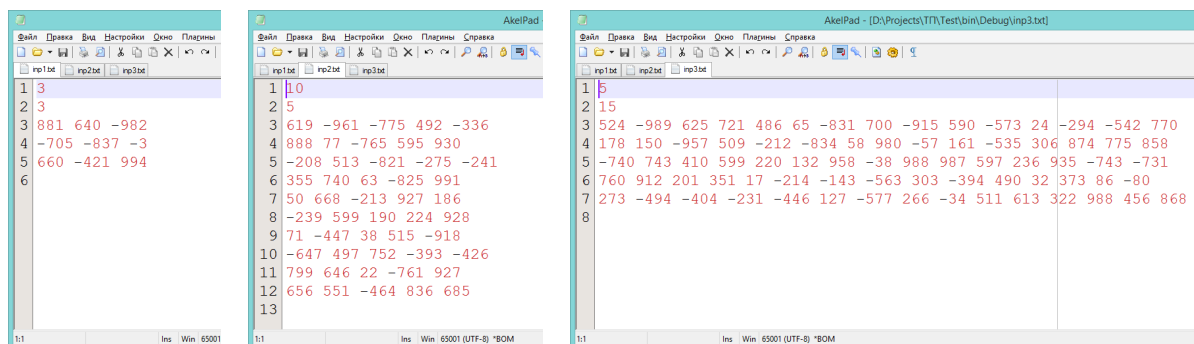


Рисунок 5.2 – Исходные файлы, полученные автоматически с использованием программы-генератора

После разработки программы-генератора можно решить учебную задачу и протестировать ее на различных наборах данных. Исходный код показан на рисунке 5.3.

```

1  using System;
2  using System.IO;
3
4  namespace Test
5  {
6      class Program
7      {
8          static void Main(string[] args)
9          {
10             TextWriter save_out = Console.Out;
11             TextReader save_in = Console.In;
12             var new_out = new StreamWriter(@"output.txt");
13             var new_in = new StreamReader(@"inp2.txt");
14             Console.SetOut(new_out);
15             Console.SetIn(new_in);
16
17             // Чтение данных из файла и инициализация массива
18             // и сразу выводим исходную
19             int N = Convert.ToInt32(Console.ReadLine());
20             int M = Convert.ToInt32(Console.ReadLine());
21
22             Console.WriteLine("*** Input matrix ***");
23

```

```

24 int[,] mas = new int[N, M];
25 for (int i = 0; i < N; i++)
26 {
27     String str_all = Console.ReadLine();
28     string[] str_elem = str_all.Split(' ');
29     for (int j = 0; j < M; j++)
30     {
31         mas[i, j] = Convert.ToInt32(str_elem[j]);
32         Console.Write(mas[i, j] + " ");
33     }
34     Console.WriteLine();
35 }
36
37 // Ищем и выводим максимальные элементы для строк
38 // Находим среднее арифметическое
39 int max = -1001;
40 float sa = 0;
41 Console.WriteLine("*** Max row elements ***");
42 for (int i = 0; i < N; i++)
43 {
44     max = -1001;
45     for (int j = 0; j < M; j++)
46     {
47         if (mas[i, j] > max)
48             max = mas[i, j];
49         sa += mas[i, j];
50     }
51     Console.WriteLine(max);
52 }
53 sa = sa / (M*N);
54
55 // Вывод модифицированной матрицы
56 Console.WriteLine("*** Modified matrix ***");
57 Console.WriteLine(string.Format("Avg = {0:0.000}", sa));
58 for (int i = 0; i < N; i++)
59 {
60     for (int j = 0; j < M; j++)
61     {
62         if (mas[i, j] > sa)
63             Console.Write("+ ");
64         else
65             Console.Write("- ");
66     }
67     Console.WriteLine();
68 }
69
70 Console.SetOut(save_out); new_out.Close();
71 Console.SetIn(save_in); new_in.Close();
72
73 }
74
75 }

```

Рисунок 5.3 – Исходный код решения учебной задачи

Входной и выходной файл показаны на рисунке 5.4.

```

1 10
2 5
3 619 -961 -775 492 -336
4 888 77 -765 595 930
5 -208 513 -821 -275 -241
6 355 740 63 -825 991
7 50 668 -213 927 186
8 -239 599 190 224 928
9 71 -447 38 515 -918
10 -647 497 752 -393 -426
11 799 646 22 -761 927
12 656 551 -464 836 685
13

```

а)

```

1 *** Input matrix ***
2 619 -961 -775 492 -336
3 888 77 -765 595 930
4 -208 513 -821 -275 -241
5 355 740 63 -825 991
6 50 668 -213 927 186
7 -239 599 190 224 928
8 71 -447 38 515 -918
9 -647 497 752 -393 -426
10 799 646 22 -761 927
11 656 551 -464 836 685
12 *** Max row elements ***
13 619
14 930
15 513
16 991
17 927
18 928
19 515
20 752
21 927
22 836
23 *** Modified matrix ***
24 Avg = 146,300
25 + - + -
26 + - - +
27 - + - -
28 + + - +
29 - + - +
30 - + + +
31 - - + -
32 - + + -
33 + + - +
34 + + - +
35

```

б)

Рисунок 5.4 – Файлы, используемые программой в качестве источников ввода-вывода: а) входной файл; б) выходной файл

5.3.2. Индивидуальное задание

Для инициализации исходной матрицы необходимо использовать генератор случайных чисел. Следует использовать ограничение на генерируемые числа, например, ограничить значения случайно генерируемых чисел величиной 100, 10 и т.д. (ограничение определено в условии), чтобы вывод программы был более читабельным. Алгоритм решения должен работать для массивов любой величины, поэтому размер исходного массива считывается из файла.

Вариант	Задание
1	Дана исходная матрица $M \times N$ целых чисел, принадлежащих отрезку $[-10^3, 10^3]$. В выходной файл необходимо вывести следующие значения: 1) исходную матрицу; 2) среднее арифметическое для каждой строки; 3) модифицированную матрицу, в которой все четные элементы заменены на 2, а нечетные на 1.
2	Дана исходная матрица $M \times N$ целых чисел, принадлежащих отрезку $[-10^3, 10^3]$. В выходной файл необходимо вывести следующие значения: 1) исходную матрицу; 2) вывести минимальный элемент для каждой строки; 3) среднее арифметическое (sp) положительных элементов и модифицированную матрицу, в которой все элементы, меньшие sp заменены на x, а остальные – на y.
3	Дана исходная матрица $M \times N$ целых чисел, принадлежащих отрезку $[-10^3, 10^3]$. В выходной файл необходимо вывести следующие значения: 1) исходную матрицу; 2) вывести минимальный элемент для каждого столбца; 3) среднее арифметическое (sp) четных элементов и модифицированную матрицу, в которой все элементы, меньшие sp заменены на x, а нечетные на y.
4	Дана исходная матрица $M \times N$ целых чисел, принадлежащих отрезку $[-10^3, 10^3]$. В выходной файл необходимо вывести следующие значения: 1) исходную матрицу; 2) вывести среднее арифметическое для каждого столбца; 3) модифицированную матрицу, в которой все элементы, меньшие чем среднее арифметическое для данного столбца заменены на x, а оставшиеся на y.
5	Дана исходная матрица $M \times N$ целых чисел, принадлежащих отрезку $[-10^3, 10^3]$. В выходной файл необходимо вывести следующие значения: 1) исходную матрицу; 2) вывести среднее арифметическое четных элементов для каждого столбца; 3) модифицированную матрицу, в которой все элементы, меньшие чем среднее арифметическое четных элементов для данного столбца заменены на q, а оставшиеся на z.
6	Дана исходная матрица $M \times N$ целых чисел, принадлежащих отрезку $[-10^3, 10^3]$. В выходной файл необходимо вывести следующие значения: 1) исходную матрицу; 2) вывести отрицательный максимум для каждого столбца; 3) модифицированную матрицу, в которой все элементы кратные 2, заменены на 2, кратные 3 – на 3, остальные – на 0.

7	Дана исходная матрица $M \times N$ целых чисел, принадлежащих отрезку $[-10^3, 10^3]$. В выходной файл необходимо вывести следующие значения: 1) исходную матрицу; 2) сумму элементов по каждой строке; 3) среднее арифметическое элементов по каждому столбцу.
8	Дана исходная матрица $M \times N$ целых чисел, принадлежащих отрезку $[-10^3, 10^3]$. В выходной файл необходимо вывести следующие значения: 1) исходную матрицу; 2) максимальный и минимальный элементы матрицы; 3) модифицированную матрицу, в которой все элементы, меньшие чем 100 заменены на -, а оставшиеся на +.
9	Дана исходная матрица $M \times N$ целых чисел, принадлежащих отрезку $[-10^3, 10^3]$. В выходной файл необходимо вывести следующие значения: 1) исходную матрицу; 2) вывести среднее арифметическое четных элементов для каждой строки; 3) модифицированную матрицу, в которой все элементы, меньшие чем среднее арифметическое соответствующей строки заменены на q, а оставшиеся на z.
10	Дана исходная матрица $M \times N$ целых чисел, принадлежащих отрезку $[-10^3, 10^3]$. В выходной файл необходимо вывести следующие значения: 1) исходную матрицу; 2) вывести среднее арифметическое для каждого столбца; 3) модифицированную матрицу, в которой все элементы, меньшие чем среднее арифметическое соответствующего столбца заменены на x, а оставшиеся на y.
11	Дана исходная матрица $M \times N$ целых чисел, принадлежащих отрезку $[-10^3, 10^3]$. В выходной файл необходимо вывести следующие значения: 1) исходную матрицу; 2) максимальный элемент для каждого столбца; 3) модифицированную матрицу, в которой все элементы, меньшие по модулю 500 заменены на m, а оставшиеся на g.
12	Дана исходная матрица $M \times N$ целых чисел, принадлежащих отрезку $[-10^3, 10^3]$. В выходной файл необходимо вывести следующие значения: 1) исходную матрицу; 2) среднее арифметическое всех отрицательных элементов; 3) модифицированную матрицу, в которой все элементы, меньшие чем среднее арифметическое отрицательных элементов заменены на -, а оставшиеся на +.
13	Дана исходная матрица $M \times N$ целых чисел, принадлежащих отрезку $[-10^3, 10^3]$. В выходной файл необходимо вывести следующие значения: 1) исходную матрицу; 2) вывести среднее арифметическое элементов, расположенных выше главной диагонали; 3) модифицированную матрицу, в которой все элементы, меньшие чем среднее арифметическое всей матрицы на m, а оставшиеся на g.

14	Дана исходная матрица $M \times N$ целых чисел, принадлежащих отрезку $[-10^3, 10^3]$. В выходной файл необходимо вывести следующие значения: 1) исходную матрицу; 2) сумму элементов по каждой строке; 3) сумму элементов по каждому столбцу.
15	Дана исходная матрица $M \times N$ целых чисел, принадлежащих отрезку $[-10^3, 10^3]$. В выходной файл необходимо вывести следующие значения: 1) исходную матрицу; 2) минимальный элемент в каждой строке; 3) модифицированную матрицу, в которой все элементы, кратные 2 заменены на 2, кратные 5 – на 5, а оставшиеся на x.
16	Дана исходная матрица $M \times N$ целых чисел, принадлежащих отрезку $[-10^3, 10^3]$. В выходной файл необходимо вывести следующие значения: 1) исходную матрицу; 2) вывести среднее арифметическое нечетных элементов для каждого столбца; 3) модифицированную матрицу, в которой все элементы, кратные 3 заменены на 3, кратные 7 – на 7, а оставшиеся на 0.
17	Дана исходная матрица $M \times N$ целых чисел, принадлежащих отрезку $[-10^3, 10^3]$. В выходной файл необходимо вывести следующие значения: 1) исходную матрицу; 2) максимум для каждого столбца; 3) модифицированную матрицу, в которой все четные элементы заменены на 0, а оставшиеся на 1.
18	Дана исходная матрица $M \times N$ целых чисел, принадлежащих отрезку $[-10^3, 10^3]$. В выходной файл необходимо вывести следующие значения: 1) исходную матрицу; 2) среднее арифметическое четных элементов; 3) модифицированную матрицу, в которой все элементы, большие значения, вычисленного в задании 2) заменены на B, остальные – на s.
19	Дана исходная матрица $M \times N$ целых чисел, принадлежащих отрезку $[-10^3, 10^3]$. В выходной файл необходимо вывести следующие значения: 1) исходную матрицу; 2) среднее арифметическое для каждой строки; 3) модифицированную матрицу, в которой все элементы, большие соответствующего значения, вычисленного в задании 2) заменены на U, остальные – на x.
20	Дана исходная матрица $M \times N$ целых чисел, принадлежащих отрезку $[-10^3, 10^3]$. В выходной файл необходимо вывести следующие значения: 1) исходную матрицу; 2) среднее арифметическое для каждого столбца; 3) модифицированную матрицу, в которой все элементы, большие соответствующего значения, вычисленного в задании 2) заменены на L, остальные – на m.

21	Дана исходная матрица $M \times N$ целых чисел, принадлежащих отрезку $[-10^3, 10^3]$. В выходной файл необходимо вывести следующие значения: 1) исходную матрицу; 2) среднее арифметическое четных элементов для каждого столбца; 3) модифицированную матрицу, в которой все элементы, большие соответствующего значения, вычисленного в задании 2) заменены на L, остальные – на m.
22	Дана исходная матрица $M \times N$ целых чисел, принадлежащих отрезку $[-10^3, 10^3]$. В выходной файл необходимо вывести следующие значения: 1) исходную матрицу; 2) сумму элементов для каждой строки; 3) модифицированную матрицу, в которой все элементы, кратные 5 заменены на 5, остальные – на x.
23	Дана исходная матрица $M \times N$ целых чисел, принадлежащих отрезку $[-10^3, 10^3]$. В выходной файл необходимо вывести следующие значения: 1) исходную матрицу; 2) сумму элементов для каждого столбца; 3) модифицированную матрицу, в которой все элементы, кратные 10 заменены на 0, кратные 3 – на 3, остальные – на x.
24	Дана исходная матрица $M \times N$ целых чисел, принадлежащих отрезку $[-10^3, 10^3]$. В выходной файл необходимо вывести следующие значения: 1) исходную матрицу; 2) среднее арифметическое элементов для каждого столбца; 3) модифицированную матрицу, в которой все элементы, кратные 3 заменены на 3, кратные 5 – на 5, остальные – на 0.
25	Дана исходная матрица $M \times N$ целых чисел, принадлежащих отрезку $[-10^3, 10^3]$. В выходной файл необходимо вывести следующие значения: 1) исходную матрицу; 2) вывести максимальный элемент для каждого столбца; 3) среднее арифметическое (sp) нечетных элементов и модифицированную матрицу, в которой все элементы, меньшие sp заменены на x, а нечетные на y.
26	Дана исходная матрица $M \times N$ целых чисел, принадлежащих отрезку $[-10^3, 10^3]$. В выходной файл необходимо вывести следующие значения: 1) исходную матрицу; 2) максимальный отрицательный элемент для каждой строки; 3) модифицированную матрицу, в которой все элементы, большие соответствующего значения, вычисленного в задаче 2) заменены на -, остальные – на +.
27	Дана исходная матрица $M \times N$ целых чисел, принадлежащих отрезку $[-10^3, 10^3]$. В выходной файл необходимо вывести следующие значения: 1) исходную матрицу; 2) среднее арифметическое для каждой строки; 3) среднее арифметическое нечетных элементов для каждого столбца.

28	Дана исходная матрица $M \times N$ целых чисел, принадлежащих отрезку $[-10^3, 10^3]$. В выходной файл необходимо вывести следующие значения: 1) исходную матрицу; 2) сумму элементов для каждой строки; 3) сумму элементов для каждого столбца.
29	Дана исходная матрица $M \times N$ целых чисел, принадлежащих отрезку $[-10^3, 10^3]$. В выходной файл необходимо вывести следующие значения: 1) исходную матрицу; 2) среднее арифметическое положительных элементов для каждой строки; 3) среднее арифметическое отрицательных элементов для каждого столбца.
30	Дана исходная матрица $M \times N$ целых чисел, принадлежащих отрезку $[-10^3, 10^3]$. В выходной файл необходимо вывести следующие значения: 1) исходную матрицу; 2) сумму элементов для каждой строки; 3) модифицированную матрицу, в которой положительные элементы заменены на +, а отрицательные – на -.
31	Дана исходная матрица $M \times N$ целых чисел, принадлежащих отрезку $[-10^3, 10^3]$. В выходной файл необходимо вывести следующие значения: 1) исходную матрицу; 2) сумму элементов для каждой строки; 3) модифицированную матрицу, в которой элементы больше чем соответствующая сумма строки заменены на +, а меньшие – на -, остальные элементы – без изменений.
32	Дана исходная матрица $M \times N$ целых чисел, принадлежащих отрезку $[-10^3, 10^3]$. В выходной файл необходимо вывести следующие значения: 1) исходную матрицу; 2) сумму элементов для каждого столбца; 3) модифицированную матрицу, в которой элементы больше чем сумма соответствующего столбца заменены на М, а меньшие – на s, остальные элементы – без изменений.
33	Дана исходная матрица $M \times N$ целых чисел, принадлежащих отрезку $[-10^3, 10^3]$. В выходной файл необходимо вывести следующие значения: 1) исходную матрицу; 2) среднее арифметическое элементов для каждого столбца; 3) модифицированную матрицу, в которой элементы кратные 5 заменены на 5, кратные 7 – на 7, кратные 10 – на x, остальные – на o.
34	Дана исходная матрица $M \times N$ целых чисел, принадлежащих отрезку $[-10^3, 10^3]$. В выходной файл необходимо вывести следующие значения: 1) исходную матрицу; 2) среднее арифметическое элементов для каждой строки; 3) модифицированную матрицу, в которой элементы главной и второстепенной диагонали заменены на +, остальные – на o.

35	Дана исходная матрица $M \times N$ целых чисел, принадлежащих отрезку $[-10^3, 10^3]$. В выходной файл необходимо вывести следующие значения: 1) исходную матрицу; 2) минимальный элемент для каждого столбца; 3) модифицированную матрицу, в которой элементы больше 100 заменены на В+, меньшие -100 – на В-, остальные – на ().
36	Дана исходная матрица $M \times N$ целых чисел, принадлежащих отрезку $[-10^3, 10^3]$. В выходной файл необходимо вывести следующие значения: 1) исходную матрицу; 2) максимальный элемент для каждой строки; 3) модифицированную матрицу, в которой элементы больше 500 заменены на Р+, меньшие -500 – на Р-, остальные – на х.
37	Дана исходная матрица $M \times N$ целых чисел, принадлежащих отрезку $[-10^3, 10^3]$. В выходной файл необходимо вывести следующие значения: 1) исходную матрицу; 2) среднее арифметическое для каждой строки; 3) модифицированную матрицу, в которой все элементы, большие соответствующего значения, вычисленного в задании 2) заменены на U, остальные – на х.
38	Дана исходная матрица $M \times N$ целых чисел, принадлежащих отрезку $[-10^3, 10^3]$. В выходной файл необходимо вывести следующие значения: 1) исходную матрицу; 2) среднее арифметическое всех отрицательных элементов; 3) модифицированную матрицу, в которой все элементы, меньшие чем среднее арифметическое отрицательных элементов заменены на -, а оставшиеся на +.
39	Дана исходная матрица $M \times N$ целых чисел, принадлежащих отрезку $[-10^3, 10^3]$. В выходной файл необходимо вывести следующие значения: 1) исходную матрицу; 2) вывести минимальный элемент для каждой строки; 3) среднее арифметическое (sp) положительных элементов и модифицированную матрицу, в которой все элементы, меньшие sp заменены на х, а остальные – на у.
40	Дана исходная матрица $M \times N$ целых чисел, принадлежащих отрезку $[-10^3, 10^3]$. В выходной файл необходимо вывести следующие значения: 1) исходную матрицу; 2) сумму элементов для каждой строки; 3) модифицированную матрицу, в которой все элементы кратные 3 заменены на III, кратные 5 – на V, кратные 10 – на х, кратные 100 – на с.

5.4. Контрольные вопросы

1. Приведите пример объявления массива. Приведите пример инициализации элементов массива.

2. Что будет выведено в консоль в результате работы приложения?

```
static void Main(string[] args)
{
    int[,] dots = { { 0, 0, 0 }, { 1, 1, 1 }, { 2, 2, 2 } };

    int zz = 10;
    for (int i = 0; i < 3; i++)
    {
        zz *= dots[0, i];
    }

    Console.Write(zz);
    Console.ReadKey();
}
```

3. Что будет выведено в консоль в результате работы приложения?

```
static void Main(string[] args)
{
    int[,] dots = {
        { 10, 20, 30 },
        { 21, 17, 13 },
        { 23, 82, 82 }
    };
    Console.Write(dots.Length);
    Console.ReadKey();
}
```

4. Что будет выведено в консоль в результате работы приложения?

```
static void Main(string[] args)
{
    int[,] dots = {
        { 10, 20, 30 },
        { 40, 50, 60 },
        { 70, 80, 90 }
    };
    int[,] res = new int[3,3];
    int i = 0;
    while (i < res.Length)
    {
        res[i/3, i%3] = dots[i%3, i/3];
        i++;
    }
    Console.Write(res[1,2]);
    Console.ReadKey();
}
```

5. Приведите фрагмент кода, демонстрирующий использование генератора случайных чисел.

6. В чем преимущества и недостатки непосредственного задания значений элементов массива в { } при его объявлении? В чем недостатки такого подхода?

7. Как изменить размер объявленного и инициализированного массива (например, добавить дополнительные элементы)?

8. Что будет выведено в консоль в результате работы приложения?

```
int[,] dots = {  
    { 10, 20, 30 },  
    { 40, 50, 60 },  
    { 70, 80, 90 }  
};  
int[,] res = dots;  
int A = 0;  
for (int i = 0; i < 3; i++)  
    A += res[i, 2];  
Console.WriteLine(A);  
Console.ReadKey();
```

ЛАБОРАТОРНАЯ РАБОТА 6. КЛАССЫ. ОБЪЕКТНОЕ МОДЕЛИРОВАНИЕ

6.1. Цель и содержание

Цель лабораторной работы: изучить структуру и принципы объявления классов, освоить технологию создания экземпляров классов (объектов).

Задачи лабораторной работы:

- научиться объявлять классы;
- научиться создавать объекты классов;
- научиться работать с полями данных и методами классов.

6.2. Теоретическая часть

6.2.1. Классы и структуры

Класс – это тип данных, объединяющий данные и методы их обработки. Класс – это пользовательский шаблон, в соответствии с которым можно создавать объекты. То есть класс – это правило, по которому будет строиться объект. Сам класс не содержит данных.

Объект класса (экземпляр класса) – переменная типа класс. Объект содержит данные и методы, манипулирующие этими данными. Класс определяет, какие данные содержит объект и каким образом он ими манипулирует.

Все что справедливо для классов можно распространить и на структуры. Отличие состоит в методе хранения объектов данных типов в оперативной памяти: структуры – это типы по значению, они размещаются в стеке; классы – это ссылочные типы, объекты классов размещаются в куче. Структуры не поддерживают наследование.

Структуры применяются для представления небольших объемов данных. Объявление структур происходит с использованием ключевого слова `struct`, объявление классов – с помощью ключевого слова `class`.

Пример объявления класса:

```
// Объявление класса
public class MyFirstClass
{
    // Данные-члены класса

    // Доступные на уровне экземпляра
    public int a;
    public float b__;
    public string fio;

    // Доступные только на уровне класса
    private bool IsOK;
    private double precision;
}
```

При создании как классов, так и структур, используется ключевое слово `new`, например:

```
MyFirstClass obj = new MyFirstClass();
```

6.2.2 Структура класса

Данные и функции, объявленные внутри класса, называются членами класса (class members). Доступность членов класса может быть описана как `public`, `private`, `protected`, `internal` или `internal protected`.

3.2.1 Данные-члены

Данные-члены – это те структуры внутри класса, которые содержат данные класса – поля, константы события.

Поля – это любые переменные, ассоциированные с классом. После создания экземпляра класса к полям можно обращаться с использованием синтаксиса `ИмяПоля.ИмяОбъекта`, например: `ИмяПоля.ИмяОбъекта`.

```
MyFirstClass obj = new MyFirstClass();

obj.a = 5;
int cc = obj.a;

obj.fio = "Novak E.I.";
```

Аналогичным образом с классом ассоциируются константы.

События будут рассмотрены в следующих лабораторных работах.

3.2.2 Функции-члены

Функции-члены – это члены, которые обеспечивают некоторую функциональность для манипулирования данными классов. Они делятся на следующие виды: методы, свойства, конструкторы, финализаторы, операции и индексаторы.

Методы (method) – это функции, ассоциированные с определенным классом. Как и данные-члены, по умолчанию они являются членами экземпляра. Они могут быть объявлены статическими с помощью модификатора static.

Свойства (property) – это наборы функций, которые могут быть доступны клиенту таким же способом, как общедоступные поля класса. В С# предусмотрен специальный синтаксис для реализации чтения и записи свойств для классов, поэтому писать собственные методы с именами, начинающимися на Set и Get, не понадобится. Поскольку не существует какого-то отдельного синтаксиса для свойств, который отличал бы их от нормальных функций, создается иллюзия объектов как реальных сущностей, предоставляемых клиентскому коду.

Конструкторы (constructor) – это специальные функции, вызываемые автоматически при инициализации объекта. Их имена совпадают с именами классов, которым они принадлежат, и они не имеют типа возврата. Конструкторы полезны для инициализации полей класса.

Финализаторы (finalizer) похожи на конструкторы, но вызываются, когда среда CLR определяет, что объект больше не нужен. Они имеют то же имя, что и класс, но с предшествующим символом тильды (~). Предсказать точно, когда будет вызван финализатор, невозможно.

Операции (operator) – это простейшие действия вроде + или -. Когда вы складываете два целых числа, то, строго говоря, применяете операцию + к целым. Однако С# позволяет указать, как существующие операции будут работать с пользовательскими классами (так называемая перегрузка операций).

Индексаторы (indexer) позволяют индексировать объекты таким же способом, как массив или коллекцию.

В данной лабораторной работе рассматриваются только методы класса – это функции, ассоциированные с определенным классом.

В С# объявление метода класса состоит из спецификатора доступности, возвращаемого значения, имени метода, списка формальных параметров и тела метода:

```
[модификатор] тип_возврата имя_метода ([список_параметров])  
{  
    // тело метода  
}
```

Например, добавим методы для объявленного ранее класса MyFirstClass:

```
// Объявление класса  
public class MyFirstClass  
{  
    // Данные-члены класса  
  
    // Доступные на уровне экземпляра  
    public int a;  
    public float b__;  
    public string fio;  
  
    // Доступные только на уровне класса  
    private bool IsOK;  
    private double precision;  
  
    // Метод для инициализации некоторых полей класса  
    public void InitClassMembers(int pA, float pB__, string pFio)  
    {  
        a = pA;  
        b__ = pB__;  
        fio = pFio;  
    }  
  
    // Метод, возвращающий значение вычисленное на основе полей класса  
    public int GetAbsA()  
    {  
        return Math.Abs(a);  
    }  
}
```

Синтаксис вызова методов аналогичен синтаксису обращения к данным-членам:

```
MyFirstClass obj = new MyFirstClass();  
  
obj.InitClassMembers(10, 0.8F, "Новиков П.Е.");  
  
int abs_a = obj.GetAbsA();
```

В данном примере метод `InitClassMembers` не возвращает никаких данных, но требует передачи ему фактических параметров. В свою очередь, метод `GetAbsA` возвращает значение типа `int` и не предполагает никаких параметров.

В общем случае параметры могут передаваться методу либо по значению, либо по ссылке. Когда переменная передается по ссылке, вызываемый метод получает саму переменную, поэтому любые изменения, которым она подвергнется внутри метода, останутся в силе после его завершения. Но если переменная передается по значению, вызываемый метод получает копию этой переменной, а это значит, что все изменения в ней по завершении метода будут утеряны. Для сложных типов данных передача по ссылке более эффективна из-за большого объема данных, который приходится копировать при передаче по значению.

Если не указано обратное, то в `C#` все параметры передаются по значению. Тем не менее, можно принудительно передавать значения по ссылке, для чего используется ключевое слово `ref`. Если параметр передается в метод, и входной аргумент этого метода снабжен префиксом `ref`, то любые изменения этой переменной, которые сделает метод, отразятся на исходном объекте.

В `C`-подобных языках функции часто возвращают более одного значения. Это обеспечивается применением выходных параметров, за счет присваивания значений переменным, переданным в метод по ссылке. Часто первоначальное значение таких переменных не важно. Эти значения перезаписываются в функции, которая может даже не обращать внимания на то, что в них хранилось первоначально.

Было бы удобно использовать то же соглашение в `C#`. Однако в `C#` требуется, чтобы переменные были инициализированы каким-то начальным значением перед тем, как к ним будет выполнено обращение. Хотя можно инициализировать входные переменные какими-то бессмысленными значениями до передачи их в функцию, которая наполнит их осмысленными значениями, этот прием выглядит в лучшем случае излишним, а в худшем – сбивающим с толку. Тем не менее, существует способ обойти требование компилятора `C#` относительно начальной инициализации переменных.

Это достигается ключевым словом `out`. Когда входной аргумент снабжен префиксом `out`, этому методу можно передать неинициализированную переменную. Переменная передается по ссылке, поэтому любые изменения, выполненные методом в переменной, сохраняются после того, как он вернет управление. Ключевое слово `out` также должно указываться при вызове метода – так же, как при его определении.

Частичные классы.

Ключевое слово `partial` (частичный) позволяет определить класс, структуру или интерфейс, распределенный по нескольким файлам. Но ситуации, когда множеству разработчиков требуется доступ к одному и тому же классу, или же в ситуации, когда некоторый генератор кода генерирует часть класса, такое разделение класса на несколько файлов может оказаться полезным. Ключевое слово `partial` просто помещается перед классом, структурой или интерфейсом.

Статические классы.

Статический класс функционально представляет собой то же самое, что и класс с приватным статическим конструктором. Создать экземпляр такого класса невозможно. Если указать ключевое слово `static` в объявлении класса, компилятор будет гарантировать, что к этому классу никогда не будут добавлены нестатические члены.

Класс `Object`.

Классы `.NET` изначально унаследованы от `System.Object`. Фактически, если при определении нового класса базовый класс не указан, компилятор автоматически предполагает, что он наследуется от `Object`.

Практическое значение этого в том, что помимо методов и свойств, которые программист определяет самостоятельно, также появляется доступ к множеству общедоступных и защищенных методов-членов, которые определены в классе `Object`. Эти методы присутствуют во всех определяемых классах.

6.3. Методика и порядок выполнения работы

6.3.1. Учебная задача

Условие. Разработать класс для представления объекта «Прямоугольный параллелепипед». Реализуйте все необходимые поля данных (закрытые) и методы позволяющие:

- считывать из файла состояние полей данных объекта;
- вычислять объем прямоугольного параллелепипеда;
- вычислять площадь поверхности прямоугольного параллелепипеда;
- выводить полную информацию об объекте в выходной файл.

Решение. Решение данной задачи состоит из следующих этапов: 1) проектирование класса; 2) объявление класса (кодирование); 3) демонстрация использования класса (создание объекта и вызов методов).

Проектирование класса можно производить с использованием любого средства проектирования (UML-редактор, онлайн средства построения диаграмм (<https://www.draw.io>), обычный графический редактор). На рисунке 6.1 показана диаграмма классов для спроектированного класса Par.

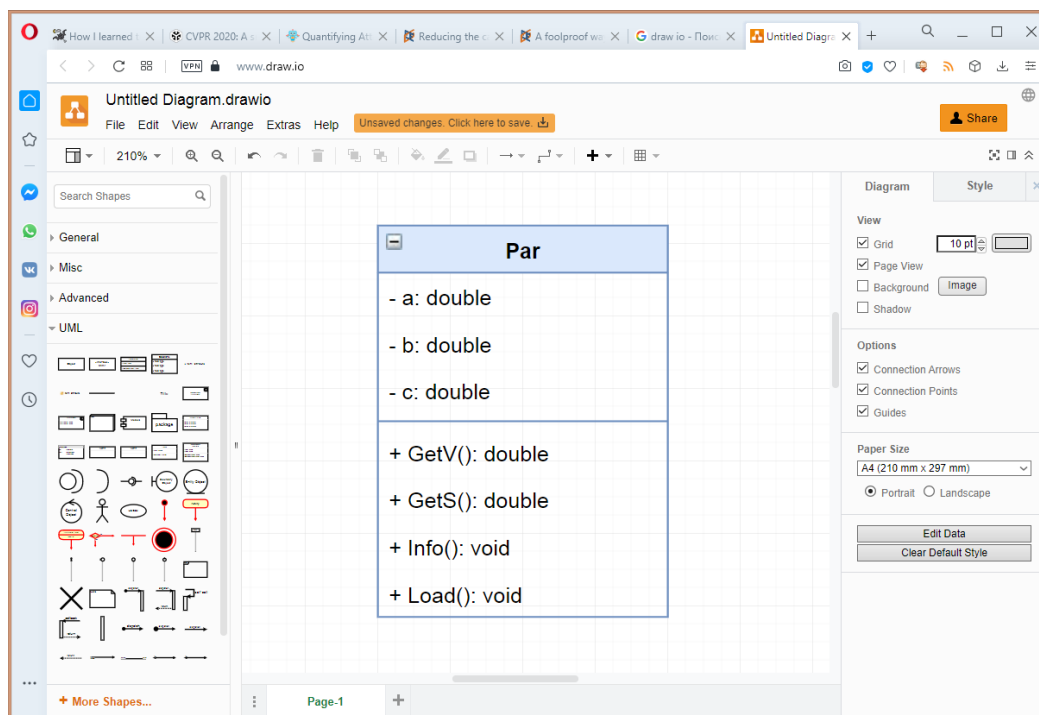


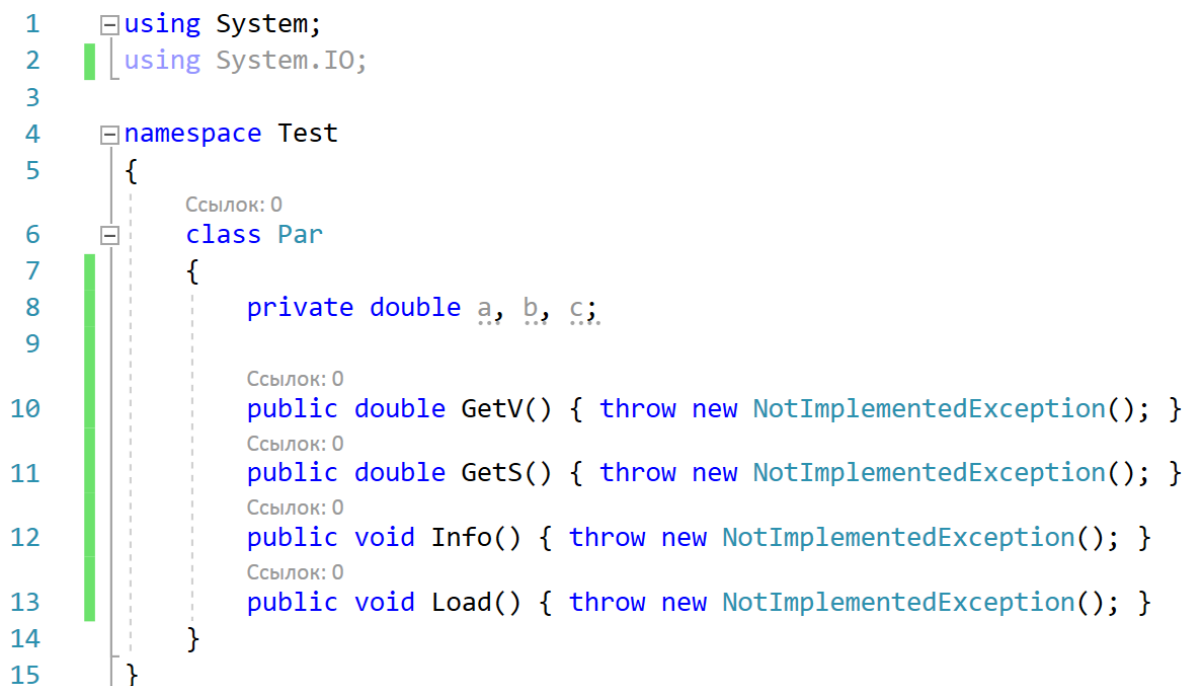
Рисунок 6.1 – Класс Par в редакторе диаграмм классов

Проект класса учитывает все данные и методы, которые можно выявить, проанализировав задание:

1) в классе определены три закрытые переменные *a*, *b*, *c*, описывающие стороны прямоугольного параллелограмма;

2) в классе должны быть следующие открытые методы: *GetV()* – возвращает объем прямоугольного параллелограмма; *GetS()* – возвращает площадь поверхности прямоугольного параллелограмма; *Info()* – выводит в стандартный поток информацию о параллелограмме; *Load()* – производит загрузку данных из входного потока для инициализации закрытых полей.

Переходим к кодированию. Добавляем новый класс *Par* в проект *Visual Studio*, наполняем его полями данных и методами (пока пустыми, позже реализуем). Листинг данного класса с методами-заглушками представлен на рисунке 6.2.



```
1  using System;
2  using System.IO;
3
4  namespace Test
5  {
6      class Par
7      {
8          private double a, b, c;
9
10         public double GetV() { throw new NotImplementedException(); }
11         public double GetS() { throw new NotImplementedException(); }
12         public void Info() { throw new NotImplementedException(); }
13         public void Load() { throw new NotImplementedException(); }
14     }
15 }
```

Рисунок 6.2 – «Набросок» класса *Par* в *Visual Studio*

Представленный набросок класса содержит пустые методы, но в проектах на любом языке нельзя оставлять методы-заглушки пустыми, или возвращающими нулевые значения. В большом объеме кода можно забыть реализовать пустой метод и возникнет трудноуловимая ошибка. Поэтому в теле метода-заглушки вставляют вызов исключения **throw new NotImplementedException()**. При попытке вызова

такого метода программа сгенерирует исключение, и программа завершится аварийно.

После этого можно наполнить класс содержанием. На рисунке 6.3 показан листинг класса, содержащий определение всех методов.

```
1  using System;
2
3  namespace Test
4  {
5      class Par
6      {
7          private double a, b, c;
8
9          public double GetV() { return a * b * c; }
10
11         public double GetS() { return 2 * (a*b + b*c + a*c); }
12
13         public void Info() {
14             String str =
15                 "*****\n" +
16                 "*" +
17                 "      параллелепипед" +
18                 "*" +
19                 "*****\n";
20             Console.WriteLine(str);
21             Console.WriteLine(string.Format(
22                 "A = {0:0.00}, B = {0:0.00}, C = {0:0.00}", a, b, c));
23             Console.WriteLine(string.Format("V = {0:0.00}", GetV()));
24             Console.WriteLine(string.Format("S = {0:0.00}", GetS()));
25         }
26
27         public void Load() {
28             a = Convert.ToDouble(Console.ReadLine());
29             b = Convert.ToDouble(Console.ReadLine());
30             c = Convert.ToDouble(Console.ReadLine());
31         }
32     }
33 }
```

Рисунок 6.3 – Определение класса Par

Теперь можно продемонстрировать использование класса Par. В данной работе необходимо создать две версии исполняемого файла: Release и Debug. Версия Release будет работать с файлами par_input.txt и output.txt в качестве источников ввода-вывода. Версия Debug будет использовать в качестве источников

стандартных потоков клавиатуру и консоль (что установлено по умолчанию). На рисунке 6.4 представлен листинг, демонстрирующий работу с классом Par.

```
1  using System;
2  using System.IO;
3
4  namespace Test
5  {
6      class Program
7      {
8          static void Main(string[] args)
9          {
10             #if !DEBUG
11                 TextWriter save_out = Console.Out;
12                 TextReader save_in = Console.In;
13                 var new_out = new StreamWriter(@"par_output.txt");
14                 var new_in = new StreamReader(@"par_input.txt");
15                 Console.SetOut(new_out);
16                 Console.SetIn(new_in);
17             #endif
18
19             Par p = new Par();
20             p.Load();
21             p.Info();
22
23             #if !DEBUG
24                 Console.SetOut(save_out); new_out.Close();
25                 Console.SetIn(save_in); new_in.Close();
26             #else
27                 Console.ReadKey();
28             #endif
29         }
30     }
31 }
```

Рисунок 6.4 – Использование класса Par

Следует обратить внимание на использование директивы препроцессора `#if`, `#else`, `#endif`. Данные директивы реализуют компиляцию кода, только если определен соответствующий идентификатор (в данном случае `DEBUG`). В данном случае используется конструкция:

```
#if !DEBUG
...
#endif
```

Это значит, что если «не DEBUG» (т.е. RELEASE), то код заключенный внутри условного блока будет компилироваться, в противном случае – нет. Переключить режим сборки можно в настройках сборки конфигурации (рисунок 6.5).

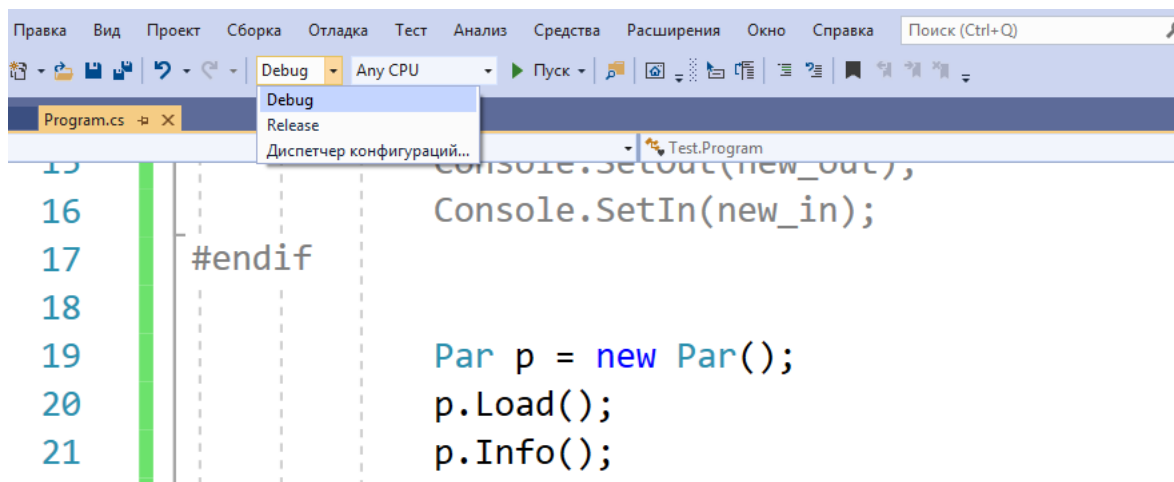


Рисунок 6.5 – Настройки конфигурации

Visual Studio создает две сборки: одна находится в папке Release, вторая – в Debug, соответственно одна программа Release – для заказчика будет работать с файлами, вторая Debug – для отладки работает с консолью.

На рисунке 6.6 показана программа Debug, на рисунке 6.7 представлены входной и выходной файл программы, собранной как Release

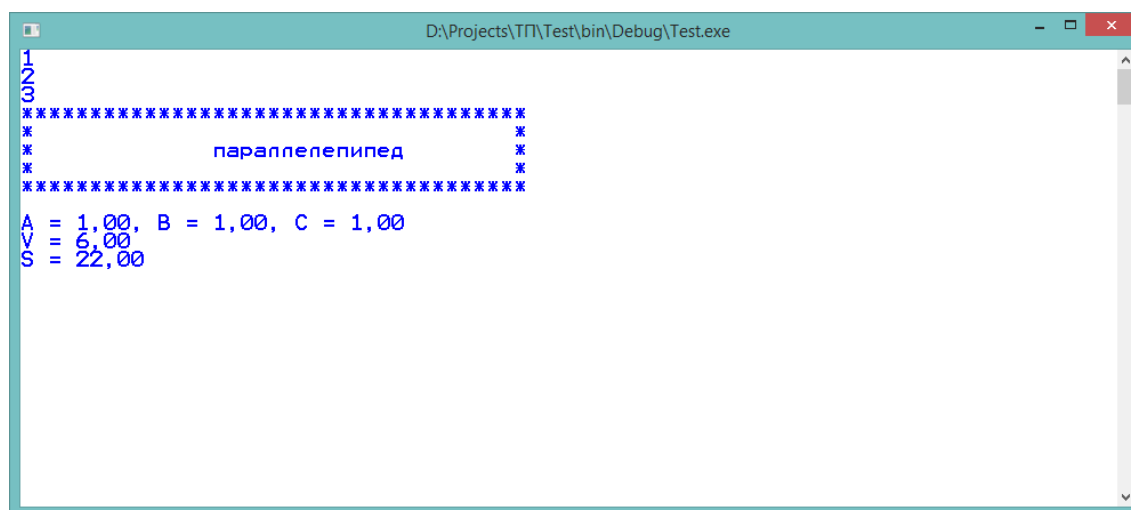


Рисунок 6.6 – Выполнение debug-приложения

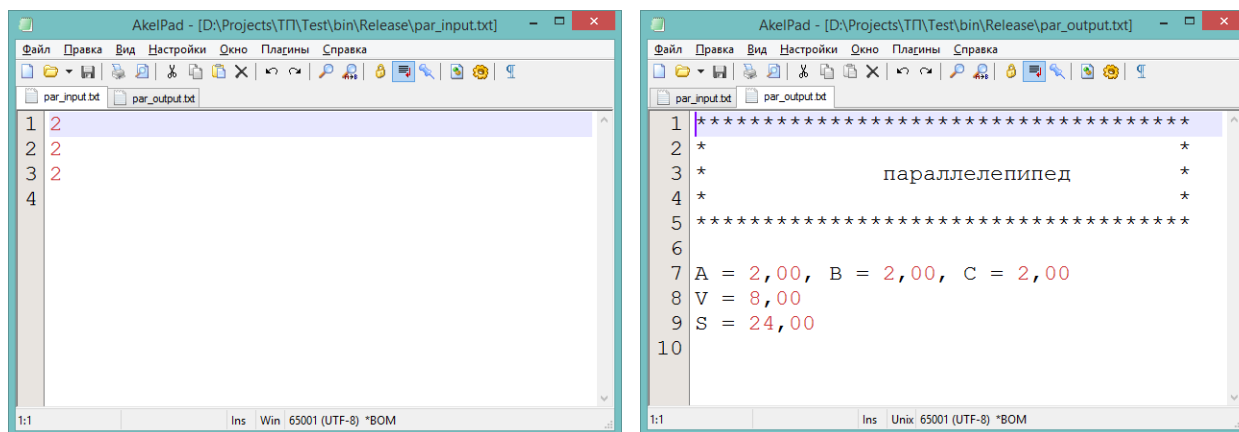


Рисунок 6.7 – Файлы release-приложения

Таким образом, в версии Debug можно реализовать довольно сложный диалог с пользователем, отладчиком, тестировщиком, а в версии Release программа будет обрабатывать данные из файлов в фоновом режиме.

6.3.2. Индивидуальное задание

Спроектируйте класс, наполните его требуемой функциональностью, продемонстрируйте работоспособность класса. Реализуйте две версии: debug и release, продемонстрируйте различное поведение программы для различных конфигураций сборки.

Вариант	Выражение для вычисления
1	Класс «Шар». Реализовать ввод и вывод полей данных, вычисление объема, диаметра и площади поверхности, а также вывод информации об объекте.
2	Класс «Куб». Реализовать ввод и вывод полей данных, вычисление объема, площади поверхности, длины диагонали, а также вывод информации об объекте.
3	Класс «Сфера». Реализовать ввод и вывод полей данных, вычисление объема, диаметра и площади поверхности, а также вывод информации об объекте.
4	Класс «Точка в пространстве». Реализовать ввод и вывод полей данных, вычисление расстояния до введенной пользователем точки, расстояния от начала координат, а также вывод информации об объекте.
5	Класс «График $y=x$ ». Реализовать ввод и вывод полей данных, вычисление интеграла функции от a до b (вводятся пользователем), длины отрезка функции от $(a, y(a))$ до $(b, y(b))$, а также вывод информации об объекте.

6	Класс «Шар». Реализовать ввод и вывод полей данных, вычисление объема, диаметра и площади поверхности, а также вывод информации об объекте.
7	Класс «Матрица $M \times N$ ». Реализовать инициализацию элементов матрицы случайными числами, вывод матрицы, нахождение максимального и минимального элементов, а также вывод информации об объекте.
8	Класс «Прямоугольный треугольник». Реализовать ввод и вывод полей данных, вычисление гипотенузы, площади и периметра, а также вывод информации об объекте.
9	Класс «Отрезок». Реализовать ввод и вывод полей данных (координаты начала и координаты конца отрезка), вычисление длины, расстояний начала и конца отрезка от начала координат, а также вывод информации об объекте.
10.	Класс «Цилиндр». Реализовать ввод и вывод полей данных, вычисление объема, площади поверхности, а также вывод информации об объекте.
11	Класс «Ромб». Реализовать ввод и вывод полей данных (диагонали ромба), вычисление площади, периметра, а также вывод информации об объекте.
12	Класс «Точка в пространстве». Реализовать ввод и вывод полей данных, вычисление расстояния до введенной пользователем точки, расстояния от начала координат, а также вывод информации об объекте.
13	Класс «График $y=3x+5$ ». Реализовать ввод и вывод полей данных, вычисление интеграла функции от a до b (вводятся пользователем), длины отрезка функции от $(a, y(a))$ до $(b, y(b))$, а также вывод информации об объекте.
14	Класс «Матрица $M \times N$ ». Реализовать инициализацию элементов матрицы случайными числами, вывод транспонированной матрицы, нахождение среднего арифметического всех элементов, а также вывод информации об объекте.
15	Класс «Отрезок в пространстве». Реализовать ввод и вывод полей данных (координаты начала и координаты конца отрезка), вычисление длины, расстояний начала и конца отрезка от начала координат, а также вывод информации об объекте.
16	Класс «График $y=x-10$ ». Реализовать ввод и вывод полей данных, вычисление интеграла функции от a до b (вводятся пользователем), длины отрезка функции от $(a, y(a))$ до $(b, y(b))$, а также вывод информации об объекте.
17	Класс «Матрица $M \times N$ ». Реализовать инициализацию элементов матрицы случайными числами, вывод транспонированной матрицы, нахождение и вывод среднего арифметического элементов в каждом столбце.

18	Класс «Куб». Реализовать ввод и вывод полей данных, вычисление объема, площади поверхности, длины диагонали, а также вывод информации об объекте.
19	Класс «Сфера». Реализовать ввод и вывод полей данных, вычисление объема, диаметра и площади поверхности, а также вывод информации об объекте.
20	Класс «Точка в пространстве». Реализовать ввод и вывод полей данных, вычисление расстояния до введенной пользователем точки, расстояния от начала координат, а также вывод информации об объекте.
21	Класс «График $y=x$ ». Реализовать ввод и вывод полей данных, вычисление интеграла функции от a до b (вводятся пользователем), длины отрезка функции от $(a, y(a))$ до $(b, y(b))$, а также вывод информации об объекте.
22	Класс «Точка в пространстве». Реализовать ввод и вывод полей данных, вычисление расстояния до введенной пользователем точки, расстояния от начала координат, а также вывод информации об объекте.
23	Класс «График $y=-x$ ». Реализовать ввод и вывод полей данных, вычисление интеграла функции от a до b (вводятся пользователем), длины отрезка функции от $(a, y(a))$ до $(b, y(b))$, а также вывод информации об объекте.
24	Класс «Цилиндр». Реализовать ввод и вывод полей данных, вычисление объема, площади поверхности, а также вывод информации об объекте.
25	Класс «Отрезок в пространстве». Реализовать ввод и вывод полей данных (координаты начала и координаты конца отрезка), вычисление длины, расстояний начала и конца отрезка от начала координат, а также вывод информации об объекте.
26	Класс «Правильная треугольная пирамида». Реализовать ввод и вывод полей данных, вычисление объема, площади поверхности, а также вывод информации об объекте.
27	Класс «Куб». Реализовать ввод и вывод полей данных, вычисление объема, площади поверхности, а также вывод информации об объекте.
28	Класс «Конус». Реализовать ввод и вывод полей данных, вычисление объема, площади поверхности, а также вывод информации об объекте.
29	Класс «Усеченный конус». Реализовать ввод и вывод полей данных, вычисление объема, площади поверхности, а также вывод информации об объекте.
30	Класс «Четырехугольная пирамида». Реализовать ввод и вывод полей данных, вычисление объема, площади поверхности, а также вывод информации об объекте.

31	Класс «Шар». Реализовать ввод и вывод полей данных, вычисление объема, площади поверхности, а также вывод информации об объекте.
32	Класс «Круг». Реализовать ввод и вывод полей данных, вычисление длины окружности, площади, а также вывод информации об объекте.
33	Класс «Параллелограмм». Реализовать ввод и вывод полей данных, вычисление периметра, площади, а также вывод информации об объекте.
34	Класс «Прямоугольный треугольник». Реализовать ввод и вывод полей данных, вычисление площади, гипотенузы, а также вывод информации об объекте.
35	Класс «Товар» с полями цена, количество, скидка. Реализовать ввод и вывод полей данных, вычисление суммарной стоимости со скидкой и без, а также вывод информации об объекте.
36	Класс «Точка на плоскости». Реализовать ввод и вывод полей данных, вычисление расстояния до начала координат, расстояния до заданной пользователем точки, а также вывод информации об объекте.
37	Класс «Книга» с полями: название, автор, количество страниц, количество строк на странице, количество символов в строке. Реализовать ввод и вывод полей данных, вычисление количества строк во всей книге, количества символов во всей книге, а также вывод информации об объекте.
38	Класс «Транспорт» с полями: скорость, объем бака, расход топлива на 1 километр. Реализовать ввод и вывод полей данных, вычисление времени в пути по переданному расстоянию, количества затрачиваемого топлива по переданному расстоянию, хватит или нет топлива на заданное расстояние, а также вывод информации об объекте.
39	Класс «Тест», содержащий поля: тема, вопрос, ответ. Реализовать ввод и вывод полей данных, функционал вывода вопроса, получения ответа, вывода информации верен ответ или нет, а также вывод информации об объекте.
40	Класс «Равноускоренное движение», содержащий поля начальная скорость, ускорение. Реализовать ввод и вывод полей данных, вычисление пройденного расстояния за заданное время, скорость в заданный момент времени после начала движения, а также вывод информации об объекте.

6.4. Контрольные вопросы

1. Приведите определение понятия «класс».
2. Что такое структура? Чем структура отличается от класса?
3. Чем отличаются определения классов?

<pre>class P { int a, b, c; }</pre>	<pre>class P { private int a, b, c; }</pre>
---	---

4. Опишите ошибки компиляции (если есть) в представленном фрагменте:

Ссылка: 2

```
class Person {  
    private int a, b, c;  
}
```

Ссылка: 0

```
static void Main(string[] args)  
{  
    Person p = new Person();  
    p.a = 100;  
    p.b = 200;  
    p.c = 300;  
}
```

4. Опишите ошибки компиляции (если есть) в представленном фрагменте:

Ссылка: 3

```
class Person {  
    public int a, b, c;  
    ссылка: 1  
    public Person(int pA, int pB, int pC)  
    {  
    }  
}
```

Ссылка: 0

```
static void Main(string[] args)  
{  
    Person p = new Person();  
    p.a = 100;  
    p.b = 200;  
    p.c = 300;  
}
```

5. Как называется переменная типа класс?
6. С использованием какого зарезервированного слова происходит объявление класса? Какое зарезервированное слово используется для объявления структуры?

7. Опишите ошибки компиляции (если есть) в представленном фрагменте:

```
public class E1
{
    int Z;

    public E1(int pParam)
    {
        Z = pParam;
    }

    float p;
    public int e;

    void Print() { Console.WriteLine(p.ToString()); }
}
```

8. Какие члены класса являются закрытыми?

```
public class E1
{
    int A;
    float B;
    public int C;
    private float D;
}
```

9. Какие поля объявленного класса доступны из объектной переменной?

```
public class Point
{
    protected int x;
    int y;
}
```

10. К каким полям нельзя обратиться из объектной переменной?

```
class Person {
    public int arc;
    private float boo;
    protected int cc;
    char simbol;
    Ссылка: 0
    public string answer { get; set; }
}
```

ЛАБОРАТОРНАЯ РАБОТА 7. КОНСТРУКТОР КЛАССА. ПЕРЕГРУЗКА МЕТОДОВ КЛАССА.

7.1. Цель и содержание

Цель лабораторной работы: понять принципы работы конструктора.

Задачи лабораторной работы:

- научиться объявлять конструктор класса;
- научиться создавать перегруженные конструкторы.

7.2. Теоретическая часть

Конструктор – это метод класса, который не возвращает значения и имеет то же самое имя, что и класс. Если конструктор класса не определен программистом явно, то компилятор создаст конструктор по умолчанию.

Конструкторы подчиняются тем же правилам перегрузки, что и все методы.

В С# поддерживается перегрузка методов – то есть может существовать несколько версий одного метода, но с разными сигнатурами (методы отличаются количеством и / или типом параметров). Чтобы перегрузить метод, просто объявляются методы с одинаковыми именами, но разными сигнатурами.

7.3. Методика и порядок выполнения работы

7.3.1. Учебная задача

Условие. Доработать класс, спроектированный в предыдущей лабораторной работе, следующим образом:

- 1) обеспечить перегрузку методов класса (2-3 перегруженных метода);
- 2) обеспечить перегрузку конструктора (2-3 перегруженных конструктора).

Решение. Для выполнения данного задания добавим к классу `Par` (рисунок 6.3) следующие методы (рисунок 7.1):

```

1      using System;
2
3      namespace Test
4      {
5          Ссылка: 7
6          class Par
7          {
8              private double a, b, c;
9
10             Ссылка: 0
11             private Par() { }
12
13             Ссылка: 2
14             public Par(double pA, double pB, double pC)
15             {
16                 a = pA; b = pB; c = pC;
17             }
18
19             Ссылка: 0
20             private void Load()
21             {
22                 a = Convert.ToDouble(Console.ReadLine());
23                 b = Convert.ToDouble(Console.ReadLine());
24                 c = Convert.ToDouble(Console.ReadLine());
25             }
26
27             public static Par CreateParFromFile()
28             {
29                 double aa = Convert.ToDouble(Console.ReadLine());
30                 double bb = Convert.ToDouble(Console.ReadLine());
31                 double cc = Convert.ToDouble(Console.ReadLine());
32                 return new Par(aa, bb, cc);
33             }
34
35             ссылка: 1
36             public double GetV() { return a * b * c; }
37
38             ссылка: 1
39             public double GetS() { return 2 * (a * b + b * c + a * c); }
40
41             Ссылка: 3
42             public void Info()
43             {
44                 String str =
45                     "\n*****\n" +
46                     "*\n" +
47                     "*\n" +
48                     "*\n" +
49                     "*\n" +
50                     "*****";
51
52                 Console.WriteLine(str);
53                 Console.WriteLine(string.Format(
54                     "A = {0:0.00}, B = {0:0.00}, C = {0:0.00}", a, b, c));
55                 Console.WriteLine(string.Format("V = {0:0.00}", GetV()));
56                 Console.WriteLine(string.Format("S = {0:0.00}", GetS()));
57             }
58         }
59     }

```

```

48
49     Ссылка: 0
50     public void Info(ConsoleColor fg, ConsoleColor bgc)
51     {
52         Console.ForegroundColor = fg;
53         Console.BackgroundColor = bgc;
54         Console.Clear();
55         Info();
56     }
57 }

```

Рисунок 7.1 – Доработка класса Par

На рисунке 7.2 демонстрируется использование модифицированного класса:

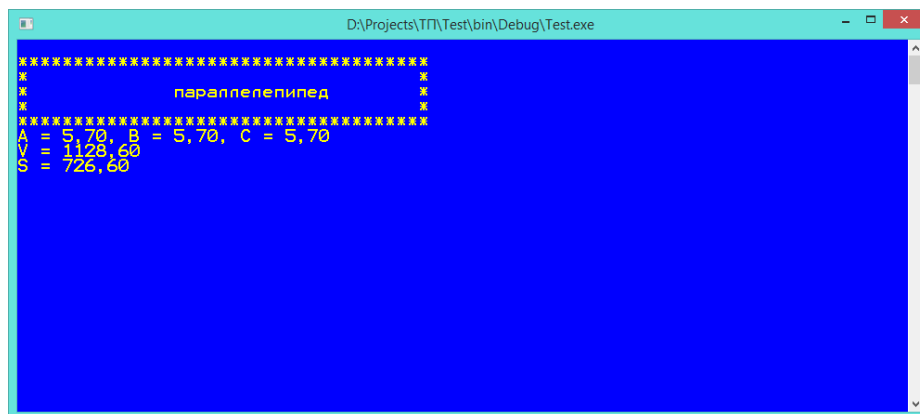
```

1  using System;
2  using System.IO;
3
4  namespace Test
5  {
6      Ссылка: 0
7      class Program
8      {
9          Ссылка: 0
10         static void Main(string[] args)
11         {
12             #if !DEBUG
13                 TextWriter save_out = Console.Out;
14                 TextReader save_in = Console.In;
15                 var new_out = new StreamWriter(@"par_output.txt");
16                 var new_in = new StreamReader(@"par_input.txt");
17                 Console.SetOut(new_out);
18                 Console.SetIn(new_in);
19             #endif
20             Par p1, p2;
21             #if DEBUG
22                 p2 = new Par(5.7, 11, 18);
23                 p2.Info(ConsoleColor.Yellow, ConsoleColor.Blue);
24             #endif
25             #if !DEBUG
26                 p1 = Par.CreateParFromFile();
27                 p1.Info();
28                 p2 = new Par(5.6, 5.11, 10.5);
29                 p2.Info();
30             #endif
31             #if !DEBUG
32                 Console.SetOut(save_out); new_out.Close();
33                 Console.SetIn(save_in); new_in.Close();
34             #endif
35             #if DEBUG
36                 Console.ReadKey();
37             #endif
38         }
39     }
40 }

```

Рисунок 7.2 – Использование класса Par с перегруженными методами

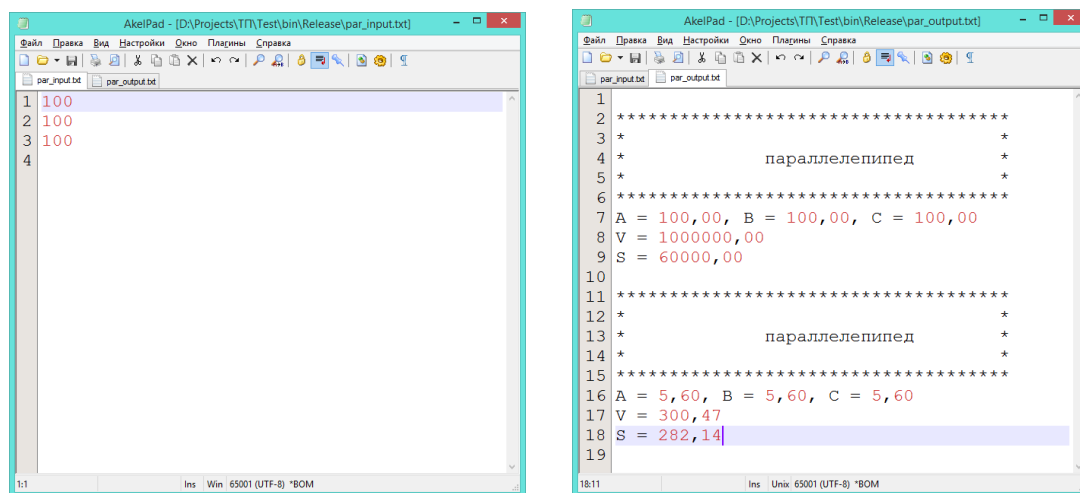
На рисунке 7.3 представлен результат работы debug-программы.



```
*****
*                                     *
*               параллелепипед       *
*                                     *
*****
A = 5,70, B = 5,70, C = 5,70
V = 1128,60
S = 726,60
```

Рисунок 7.3 – Выполнение debug-приложения

На рисунке 7.4 показан входной и выходной файл с которыми работает release-приложение.



```
1 100
2 100
3 100
4

1 *****
2 *                                     *
3 *               параллелепипед       *
4 *                                     *
5 *
6 *****
7 A = 100,00, B = 100,00, C = 100,00
8 V = 1000000,00
9 S = 60000,00
10
11 *****
12 *                                     *
13 *               параллелепипед       *
14 *                                     *
15 *
16 A = 5,60, B = 5,60, C = 5,60
17 V = 300,47
18 S = 282,14
19
```

Рисунок 7.4 – Файлы release-приложения

7.3.2. Индивидуальное задание

Условие. Спроектировать и реализовать класс в соответствии с вариантом, с учетом следующих требований:

- 1) обеспечить перегрузку методов класса (2-3 перегруженных метода);
- 2) обеспечить перегрузку конструктора (2-3 перегруженных конструктора);
- 3) различное поведение приложения в debug и release сборках.

Вариант	Выражение для вычисления
1	Класс «График $y=x$ ». Реализовать ввод и вывод полей данных, вычисление интеграла функции от a до b (вводятся пользователем), длины отрезка функции от $(a, y(a))$ до $(b, y(b))$, а также вывод информации об объекте.
2	Класс «Точка в пространстве». Реализовать ввод и вывод полей данных, вычисление расстояния до введенной пользователем точки, расстояния от начала координат, а также вывод информации об объекте.
3	Класс «График $y=-x$ ». Реализовать ввод и вывод полей данных, вычисление интеграла функции от a до b (вводятся пользователем), длины отрезка функции от $(a, y(a))$ до $(b, y(b))$, а также вывод информации об объекте.
4	Класс «Цилиндр». Реализовать ввод и вывод полей данных, вычисление объема, площади поверхности, а также вывод информации об объекте.
5	Класс «Отрезок в пространстве». Реализовать ввод и вывод полей данных (координаты начала и координаты конца отрезка), вычисление длины, расстояний начала и конца отрезка от начала координат, а также вывод информации об объекте.
6	Класс «Правильная треугольная пирамида». Реализовать ввод и вывод полей данных, вычисление объема, площади поверхности, а также вывод информации об объекте.
7	Класс «Куб». Реализовать ввод и вывод полей данных, вычисление объема, площади поверхности, а также вывод информации об объекте.
8	Класс «Конус». Реализовать ввод и вывод полей данных, вычисление объема, площади поверхности, а также вывод информации об объекте.
9	Класс «Усеченный конус». Реализовать ввод и вывод полей данных, вычисление объема, площади поверхности, а также вывод информации об объекте.
10.	Класс «Четырехугольная пирамида». Реализовать ввод и вывод полей данных, вычисление объема, площади поверхности, а также вывод информации об объекте.
11	Класс «Шар». Реализовать ввод и вывод полей данных, вычисление объема, площади поверхности, а также вывод информации об объекте.
12	Класс «Круг». Реализовать ввод и вывод полей данных, вычисление длины окружности, площади, а также вывод информации об объекте.
13	Класс «Параллелограмм». Реализовать ввод и вывод полей данных, вычисление периметра, площади, а также вывод информации об объекте.

14	Класс «Прямоугольный треугольник». Реализовать ввод и вывод полей данных, вычисление площади, гипотенузы, а также вывод информации об объекте.
15	Класс «Товар» с полями цена, количество, скидка. Реализовать ввод и вывод полей данных, вычисление суммарной стоимости со скидкой и без, а также вывод информации об объекте.
16	Класс «Точка на плоскости». Реализовать ввод и вывод полей данных, вычисление расстояния до начала координат, расстояния до заданной пользователем точки, а также вывод информации об объекте.
17	Класс «Книга» с полями: название, автор, количество страниц, количество строк на странице, количество символов в строке. Реализовать ввод и вывод полей данных, вычисление количества строк во всей книге, количества символов во всей книге, а также вывод информации об объекте.
18	Класс «Транспорт» с полями: скорость, объем бака, расход топлива на 1 километр. Реализовать ввод и вывод полей данных, вычисление времени в пути по переданному расстоянию, количества затрачиваемого топлива по переданному расстоянию, хватит или нет топлива на заданное расстояние, а также вывод информации об объекте.
19	Класс «Тест», содержащий поля: тема, вопрос, ответ. Реализовать ввод и вывод полей данных, функционал вывода вопроса, получения ответа, вывода информации верен ответ или нет, а также вывод информации об объекте.
20	Класс «Равноускоренное движение», содержащий поля начальная скорость, ускорение. Реализовать ввод и вывод полей данных, вычисление пройденного расстояния за заданное время, скорость в заданный момент времени после начала движения, а также вывод информации об объекте.
21	Класс «Шар». Реализовать ввод и вывод полей данных, вычисление объема, диаметра и площади поверхности, а также вывод информации об объекте.
22	Класс «Куб». Реализовать ввод и вывод полей данных, вычисление объема, площади поверхности, длины диагонали, а также вывод информации об объекте.
23	Класс «Сфера». Реализовать ввод и вывод полей данных, вычисление объема, диаметра и площади поверхности, а также вывод информации об объекте.
24	Класс «Точка в пространстве». Реализовать ввод и вывод полей данных, вычисление расстояния до введенной пользователем точки, расстояния от начала координат, а также вывод информации об объекте.

25	Класс «График $y=x$ ». Реализовать ввод и вывод полей данных, вычисление интеграла функции от a до b (вводятся пользователем), длины отрезка функции от $(a, y(a))$ до $(b, y(b))$, а также вывод информации об объекте.
26	Класс «Шар». Реализовать ввод и вывод полей данных, вычисление объема, диаметра и площади поверхности, а также вывод информации об объекте.
27	Класс «Матрица $M \times N$ ». Реализовать инициализацию элементов матрицы случайными числами, вывод матрицы, нахождение максимального и минимального элементов, а также вывод информации об объекте.
28	Класс «Прямоугольный треугольник». Реализовать ввод и вывод полей данных, вычисление гипотенузы, площади и периметра, а также вывод информации об объекте.
29	Класс «Отрезок». Реализовать ввод и вывод полей данных (координаты начала и координаты конца отрезка), вычисление длины, расстояний начала и конца отрезка от начала координат, а также вывод информации об объекте.
30	Класс «Цилиндр». Реализовать ввод и вывод полей данных, вычисление объема, площади поверхности, а также вывод информации об объекте.
31	Класс «Ромб». Реализовать ввод и вывод полей данных (диагонали ромба), вычисление площади, периметра, а также вывод информации об объекте.
32	Класс «Точка в пространстве». Реализовать ввод и вывод полей данных, вычисление расстояния до введенной пользователем точки, расстояния от начала координат, а также вывод информации об объекте.
33	Класс «График $y=3x+5$ ». Реализовать ввод и вывод полей данных, вычисление интеграла функции от a до b (вводятся пользователем), длины отрезка функции от $(a, y(a))$ до $(b, y(b))$, а также вывод информации об объекте.
34	Класс «Матрица $M \times N$ ». Реализовать инициализацию элементов матрицы случайными числами, вывод транспонированной матрицы, нахождение среднего арифметического всех элементов, а также вывод информации об объекте.
35	Класс «Отрезок в пространстве». Реализовать ввод и вывод полей данных (координаты начала и координаты конца отрезка), вычисление длины, расстояний начала и конца отрезка от начала координат, а также вывод информации об объекте.
36	Класс «График $y=x-10$ ». Реализовать ввод и вывод полей данных, вычисление интеграла функции от a до b (вводятся пользователем), длины отрезка функции от $(a, y(a))$ до $(b, y(b))$, а также вывод информации об объекте.

37	Класс «Матрица MxN». Реализовать инициализацию элементов матрицы случайными числами, вывод транспонированной матрицы, нахождение и вывод среднего арифметического элементов в каждом столбце.
38	Класс «Куб». Реализовать ввод и вывод полей данных, вычисление объема, площади поверхности, длины диагонали, а также вывод информации об объекте.
39	Класс «Сфера». Реализовать ввод и вывод полей данных, вычисление объема, диаметра и площади поверхности, а также вывод информации об объекте.
40	Класс «Точка в пространстве». Реализовать ввод и вывод полей данных, вычисление расстояния до введенной пользователем точки, расстояния от начала координат, а также вывод информации об объекте.

7.4. Контрольные вопросы

1. Что такое статический класс? Что такое статический метод?
2. Что такое конструктор класса?
3. Что такое перегрузка методов?
4. Может ли один конструктор класса вызывать другой конструктор? Прежде чем отвечать попробуйте реализовать такой вызов в своем разработанном классе.
5. Сколько перегрузок может иметь метод класса?
6. Для каких целей следует перегружать конструктор класса?
7. Что такое сигнатура метода?
8. Какие ошибки присутствуют в объявлении класса?

```

public class Vector
{
    float x;
    float y;
    float z;

    Vector(float pX, float pY, float pZ)
    {
        x = pX; y = pY; z = pZ;
    }

    public Vector()
        : this(0, 0, 0)
    {
    }
}

```

9. Дано определение класса:

```
public class Человек
{
    int ПолныхЛет;
    string ФИО;

    public Человек(int pAge, string pFIO)
    {
        ПолныхЛет = pAge;
        ФИО = pFIO;
    }
}
```

и фрагмент использования класса в функции Main():

```
static void Main(string[] args)
{
    Человек чел = new Человек(3, "");
    чел.ПолныхЛет = 5;
    чел.ФИО = "Николаев Е.И.";
}
```

Какие ошибки компиляции существуют в коде?

10. Укажите ошибки (если есть) в объявлении класса:

```
class Person {
    private String name;
    private int age;

    public Person(String pName, int pAge)
    {
        name = pName; age = pAge;
    }

    public static String GetName()
    {
        return name;
    }
}
```

ЛАБОРАТОРНАЯ РАБОТА 8. ПРОЕКТИРОВАНИЕ ИЕРАРХИИ КЛАССОВ

8.1. Цель и содержание

Цель лабораторной работы: изучить механизм организации наследования классов.

Задачи лабораторной работы:

- научиться объявлять производные классы;
- научиться создавать иерархии классов;
- научиться использовать механизм полиморфизма.

8.2. Теоретическая часть

8.2.1. Наследование реализации

Наследование реализации (implementation inheritance) означает, что тип происходит от базового типа, получая от него все поля-члены и функции-члены.

Синтаксис наследования реализации:

```
class ПроизводныйКласс: БазовыйКласс
{
    // Данные – члены и функции – члены
}
```

Если при определении класса не указан базовый класс, то C# предполагает, что базовым классом является System.Object.

8.2.2. Создание иерархии классов

При наследовании реализации производный класс наследует реализацию каждой функции базового типа, если только в его определении не указано, что реализация функции должна быть переопределена.

Определим следующую иерархию классов (рис. 6.1) и продемонстрируем, как наследуется реализация и как переопределяются свойств и методов.

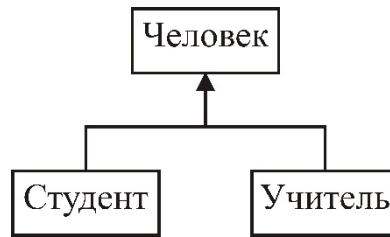


Рисунок 6.1 – Иерархия классов.

Создадим код, описывающий данную иерархию. Определим базовый класс:

```
class Человек
{
    public Человек(string Ф, string И, string О, int Возр)
    {
        Фамилия = Ф; Имя = И; Отчество = О; Возраст = Возр;
    }

    public Человек()
    {
        Фамилия = "нет данных"; Имя = ""; Отчество = "";
        Возраст = 18;
    }

    public string Фамилия, Имя, Отчество;
    private DateTime ДатаРождения;

    public virtual string ФИО
    {
        get { return Фамилия + " " + Имя + " " + Отчество; }
    }

    public int Возраст
    {
        get { return DateTime.Now.Year - ДатаРождения.Year; }
        set
        {
            int ГодРождения = DateTime.Now.Year - value;

            ДатаРождения = Convert.ToDateTime(ГодРождения.ToString()+".01.01");
        }
    }
}
```

Обратите внимание на наличие двух конструкторов, механизм хранения возраста человека и ключевое слово `virtual` у свойства «ФИО». Ключевое слово `virtual` указывает, что данное свойство будет переопределено в производном классе.

Определим производный класс «Учитель»:

```

class Учитель: Человек
{
    public Учитель()
        :base()
    {
        УченоеЗвание = УченыеЗвания.Без_Звания;
        УченаяСтепень = УченыеСтепени.Без_Степени;
    }

    public Учитель(string Ф, string И, string О, int Возр, УченыеЗвания УЗ, УченыеСтепени УС)
        :base(Ф, И, О, Возр)
    {
        УченоеЗвание = УЗ;
        УченаяСтепень = УС;
    }

    public УченыеЗвания УченоеЗвание;
    public УченыеСтепени УченаяСтепень;

    public override string ФИО
    {
        get
        {
            return УченаяСтепень.ToString() + ", " + УченоеЗвание.ToString() + ", " + base.ФИО;
        }
    }
}

```

Следует обратить внимание на использование ключевого слова `override` у свойства «ФИО», которое указывает на то, что данное свойство имеет новую реализацию, отличающуюся от реализации базового класса.

Определим второй производный класс «Студент»:

```

class Студент: Человек
{
    public Студент(string Ф, string И, string О, int Возр, Специальности Спец)
        :base(Ф, И, О, Возр)
    {
        Специальность = Спец;
    }

    public Специальности Специальность;

    public override string ФИО
    {
        get
        {
            return base.ФИО + ", " + Специальность.ToString();
        }
    }
}

```

В обоих производных классах следует обратить внимание на реализацию конструкторов производных классов, использование ключевого слова `base` в коде свойства «ФИО» и при объявлении конструкторов.

Также интерес представляют типы данных, используемые для членов-данных: «Специальности», «УченыеЗвания», «УченыеСтепени». Вот определения данных типов-перечислений:

```
public enum УченыеЗвания
{
    Доцент,
    Профессор,
    Академик,
    Без_Звания
}

public enum УченыеСтепени
{
    Кандидат_Технических_Наук,
    Кандидат_ФизМат_Наук,
    Кандидат_Педагогических_Наук,
    Доктор_ФизМат_Наук,
    Без_Степени
}

public enum Специальности
{
    Информационные_Системы_И_Технологии,
    Безопасность_Информационных_Систем,
    Технология_Защиты_Информации,
    Психология,
    Нанозлектроника
}
```

Наконец, продемонстрируем использование объявленной иерархии классов.

В программе объявим массив объектов типа «Человек». Следует понимать, что при объявлении такого массива, его элементам можно присваивать объекты любого производного класса, причем каждый объект будет вести себя по-своему (за счет переопределения свойств и методов).

8.3. Методика и порядок выполнения работы

1. Создайте консольное приложение.

2. Изучите пример создания иерархии классов, представленный в разделе «Теоретическое обоснование» данной лабораторной работы.

3. Постройте свою иерархию классов в соответствии с индивидуальным заданием. В результате выполнения лабораторной работы должны быть реализованы следующие механизмы:

- использование типа-перечисления (хотя бы одного);
- использование переопределенного свойства (хотя бы одного);
- использование переопределенного метода (хотя бы одного);
- использование вызова базового конструктора;
- использование вызова любого базового метода (отличного от конструктора).

4. Продемонстрируйте использование классов, созданной иерархии (легче всего это сделать с использованием массивов). При защите работы укажите признаки присутствия полиморфного поведения в программе (реализация полиморфизма).

Индивидуальное задание

Спроектируйте класс, наполните его требуемой функциональностью, продемонстрируйте работоспособность класса, продемонстрируйте полиморфное поведение объектов.

Вариант	Иерархия классов
1	<pre> graph BT ГрузовойАвто --> Автомобиль ЛегковойАвто --> Автомобиль Автомобиль --> ТехСредство Вертолёт --> ТехСредство </pre>
2	<pre> graph BT Сервер --> КомпТехника Ноутбук --> КомпТехника ПК --> КомпТехника </pre>
3	<pre> graph BT Квадрат --> Четырехугольник Ромб --> Четырехугольник Четырехугольник --> Геометрия Треугольник --> Геометрия </pre>
4	<pre> graph BT Книга --> Издание Журнал --> Издание ЭлРесурс --> Издание </pre>
5	<pre> graph BT Растровое --> Изображение Векторное --> Изображение Изображение --> Мультимедиа Видео --> Мультимедиа </pre>
6	<pre> graph BT Стрелковое --> Вооружение Танк --> Вооружение Сомолет --> Вооружение </pre>

Вариант	Иерархия классов
7	<pre> graph BT Машины --> Авиасредство Машины --> Грузовик Авиасредство --> Вертолет Авиасредство --> Истребитель </pre>
8	<pre> graph BT Растение --> Трава Растение --> Дерево Растение --> Кустарник </pre>
9	<pre> graph BT ПО --> ОС ПО --> Прикладное ОС --> СвободныеОС ОС --> ПлатныеОС </pre>
10	<pre> graph BT ЭВМ --> Мобильный ЭВМ --> Стационарный Мобильный --> Планшет Мобильный --> Ноутбук </pre>
11	<pre> graph BT Электроника --> СотТелефон Электроника --> Фотоаппарат Электроника --> Планшет </pre>
12	<pre> graph BT ТехСредство --> Автомобиль ТехСредство --> Вертолёт Автомобиль --> ГрузовойАвто Автомобиль --> ЛегковойАвто </pre>

Вариант	Иерархия классов
13	<pre> graph BT Телевизор --> БытТехника Утюг --> БытТехника Микроволновка --> БытТехника </pre>
14	<pre> graph BT Планшет --> Мобильный Ноутбук --> Мобильный Мобильный --> ЭВМ Стационарный --> ЭВМ </pre>
15	<pre> graph BT Ручка --> КанцТовары Органайзер --> КанцТовары Скоросшиватель --> КанцТовары </pre>
16	<pre> graph BT Вертолет --> Авиасредство Истребитель --> Авиасредство Авиасредство --> Машины Грузовик --> Машины </pre>
17	<pre> graph BT Квартира --> Недвижимость Дом --> Недвижимость Гараж --> Недвижимость </pre>
18	<pre> graph BT Трактор --> Тяжелое Комбайн --> Тяжелое Тяжелое --> Машиностроение Среднее --> Машиностроение </pre>
19	<pre> graph BT Фотон --> ЭлемЧастица Электрон --> ЭлемЧастица Протон --> ЭлемЧастица </pre>

Вариант	Иерархия классов
20	<pre> classDiagram class Геометрия class Многоугольник class Эллипс class ПрямоПараллелепипед class Треугольник Геометрия < -- Многоугольник Геометрия < -- Эллипс Многоугольник < -- ПрямоПараллелепипед Многоугольник < -- Треугольник </pre>
21	<pre> classDiagram class Специалист class Программист class Врач class Летчик Специалист < -- Программист Специалист < -- Врач Специалист < -- Летчик </pre>
22	<pre> classDiagram class ПО class ОС class Прикладное class СвободныеОС class ПлатныеОС ПО < -- ОС ПО < -- Прикладное ОС < -- СвободныеОС ОС < -- ПлатныеОС </pre>
23	<pre> classDiagram class БанкКарта class Накопительная class Платежная class Зарплатная БанкКарта < -- Накопительная БанкКарта < -- Платежная БанкКарта < -- Зарплатная </pre>
24	<pre> classDiagram class Сервис class Продажи class Лизинг class Кредит Сервис < -- Продажи Сервис < -- Лизинг Сервис < -- Кредит </pre>
25	<pre> classDiagram class Мебель class Стол class МягкМебель class Шкаф Мебель < -- Стол Мебель < -- МягкМебель Мебель < -- Шкаф </pre>

8.4. Контрольные вопросы

1. Что такое наследование реализации? Как описать синтаксически наследование реализации?
2. Для чего используется ключевое слово base?

3. Можно ли переопределить метод класса? Свойства класса? Данные класса?
4. Как переопределить метод в производном классе?
5. Для чего используется ключевое слово `virtual`?
6. Для чего используется ключевое слово `override`?
7. Как поменять цвет фона в консольном приложении?
8. Укажите ошибки (если есть) в представленном фрагменте:

```
public class Parent
{
    int a;
}

class Child : Parent
{
    int b;

    void E()
    {
        b = a;
    }
}
```

9. Сколько базовых классов может быть у любого класса в C#?
10. Даны определения классов на языке C#.

```
public class Parent
{
    protected int a;

    Parent()
    {
        a = 0;
    }
}
```

```
class Child1 : Parent
{
    int c;

    public Child1()
        :base()
    { }
}
```

Существуют ли ошибки в данном коде (укажите ошибки, если есть)?

ЛАБОРАТОРНАЯ РАБОТА 9. ПОЛИМОРФИЗМ НА ОСНОВЕ ИНТЕРФЕЙСОВ

9.1. Цель и содержание

Цель лабораторной работы: изучить принципы работы с типами интерфейсов в C#.

Задачи лабораторной работы:

- научиться объявлять интерфейсы в C#;
- научиться создавать классы, реализующие интерфейсы.

9.2. Теоретическая часть

Кроме наследования реализации (implementation inheritance) в языке C# реализован механизм наследования интерфейсов (interface inheritance). Необходимо понимать, что не все объектно-ориентированные языки поддерживают интерфейсы.

Если класс наследует интерфейс, класс как бы берет на себя обязательства реализовать некоторый функционал.

Синтаксис наследования интерфейса не отличается от синтаксиса наследования реализации:

```
class КлассРеализующийИнтерфейс: Интерфейс
{
    // Данные – члены и функции – члены
}
```

Отличие от механизма наследования реализации состоит в том, что:

- класс обязательно должен реализовать методы и свойства, продекларированные в интерфейсе (никакой реализации в интерфейсе не существует);
- класс может реализовывать (наследовать) несколько интерфейсов, тогда как базовый класс может быть только один.

Для объявления типа интерфейса используется ключевое слово `interface`:

```
public interface ICalculate
{
    void Plus(int pPlus);
    void Minus(int pMinus);
}

public interface IVisual
{
    string Name { get; set; }
    void DrawObject();
}
```

В приведенном примере объявлены два интерфейса: ICalculate и IVisual.

Интерфейс ICalculate включает два метода с одним параметром. Из названий видно, что один метод что-то увеличивает, второй – что-то уменьшает на величину параметра.

Второй интерфейс IVisual содержит метод DrawObject без параметров, который призван нарисовать объект, а также свойство Name, которое доступно как для чтения, так и для записи.

Следует понимать, что интерфейсы не предполагают конкретную реализацию методов, а свойства интерфейсов не хранят данных. Интерфейс должен быть реализован классом, который и определит конкретное наполнение методов и свойств интерфейса.

Для использования объявленных интерфейсов создадим два класса Human (человек) и Car (автомобиль).

```

public class Human
{
    public Human(string pFIO, int pAge)
    {
        FIO = pFIO;
        Age = pAge;
    }

    private string FIO;
    private int Age;
}

public class Car
{
    public Car(string pManufacturer, string pModel, int pVelocity)
    {
        Manufacturer = pManufacturer;
        Model = pModel;
        Velocity = pVelocity;
    }

    private string Manufacturer;
    private string Model;
    private int Velocity;
}

```

В каждом классе определен конструктор, который инициализирует закрытые данные-члены класса.

Требуется для каждого класса реализовать вывод в консоль, а также возможности увеличения скорости автомобиля и возможности изменения возраста у человека.

Этот общий функционал можно реализовать с использованием следующих способов:

1. Добавить все необходимые функции в каждый класс независимо. Этот метод приводит к «разбуханию кода», сложной управляемости кода.
2. Реализовать один класс, например Base (базовый), в котором определить общие методы и свойства (Plus, Minus, DrawObject). Затем использовать класс Base в качестве базового для классов Human и Car, причем в каждом классе переопределить методы базового класса с использованием ключевого слова `override`. Этот метод – наследование реализации. Получается иерархия классов, которые по смыслу и наполнению сильно отличаются. Хотя в данном подходе классы и код более упорядочены, все равно возникает избыточность за счет многократного переопределения методов.

3. Определение интерфейсов и реализация возможностей интерфейсов данными классами.

Именно метод 3 будет использован в данной лабораторной работе.

9.3. Методика и порядок выполнения работы

9.3.1 Учебная задача

1. Создайте консольное приложение.

2. Определите в приложении классы Human и Car, а также интерфейсы ICalculate и IVisual, представленные в разделе «Теоретическое обоснование» данной лабораторной работы.

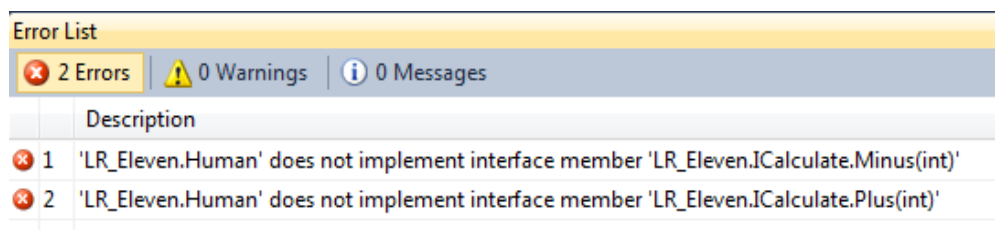
3. Реализуем механизм наследования интерфейса ICalculate классом Human.

Для этого выполним следующие действия:

3.1 В определении класса укажем, что класс Human наследует интерфейс ICalculate.

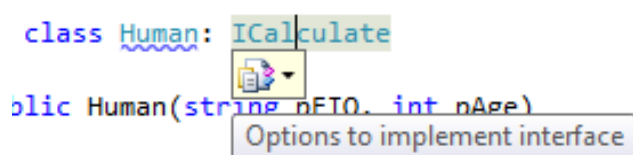
```
public class Human: ICalculate
```

3.2 Обратите внимание, что после этого попытка перекомпилировать проект приведет к ошибкам:

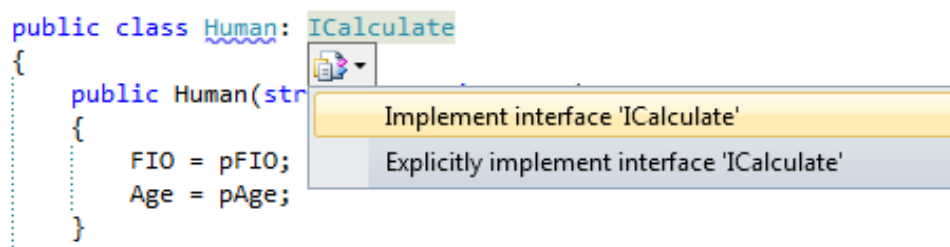


То есть компилятор сообщает, что интерфейс наследуется классом, но методы, заявленные в интерфейсе, классом не реализованы. Реализуем их.

3.3 Наведите курсор мыши на интерактивное подчеркивание и появится выпадающий список:



Раскройте его и выберите команду «Implement interface» (реализовать интерфейс).



3.3 В классе Human появятся два новых метода, как и было объявлено в интерфейсе ICalculate. Определение класса примет вид:

```
public class Human: ICalculate
{
    public Human(string pFIO, int pAge)
    {
        FIO = pFIO;
        Age = pAge;
    }

    private string FIO;
    private int Age;

    public void Plus(int pPlus)
    {
        throw new NotImplementedException();
    }

    public void Minus(int pMinus)
    {
        throw new NotImplementedException();
    }
}
```

Обратите внимание, что в каждый сгенерированный метод среда разработки добавила код вызова исключения, то есть проект скомпилируется без ошибок, но в процессе выполнения, при попытке использовать методы Plus или Minus программа завершится с ошибками. Это сделано для того, чтобы программист не забыл реализовать данные методы интерфейсов.

В процессе выполнения пп. 3.1 – 3.3 были использованы возможности Visual Studio по автоматизации реализации интерфейса. Очевидно, что интерфейс можно было реализовать, самостоятельно написав данный код.

– использование вызова базового конструктора;

– использование вызова любого базового метода (отличного от конструктора).

4. Определим окончательную реализацию для методов Plus и Minus:

```
public void Plus(int pPlus)
{
    ... Age += pPlus;
}

public void Minus(int pMinus)
{
    ... Age -= pMinus;
}
```

5. Аналогичным образом реализуем наследование интерфейса IVisual для класса Human. Окончательно для класса Human получим:

```

public class Human: ICalculate, IVisual
{
    public Human(string pFIO, int pAge)
    {
        FIO = pFIO;
        Age = pAge;
    }

    private string FIO;
    private int Age;

    public void Plus(int pPlus)
    {
        Age += pPlus;
    }

    public void Minus(int pMinus)
    {
        Age -= pMinus;
    }

    public string Name
    {
        get
        {
            return FIO + " : " + Age.ToString();
        }
        set
        {
            FIO = value;
        }
    }

    public void DrawObject()
    {
        Console.WriteLine
        (
            "      o      \n" +
            "    ----- \n" +
            "      |      \n" +
            "     /  \ \   \n" +
            "     /  \ \   \n"
        );
        Console.WriteLine(Name);
    }
}

```

6. Реализуем интерфейсы ICalculate и IVisual для класса Car:

```

public class Car: IVisual, ICalculate
{
    public Car(string pManufacturer, string pModel, int pVelocity)
    {
        Manufacturer = pManufacturer;
        Model = pModel;
        Velocity = pVelocity;
    }

    private string Manufacturer;
    private string Model;
    private int Velocity;

    public void Plus(int pPlus)
    {
        Velocity += pPlus;
    }

    public void Minus(int pMinus)
    {
        Velocity += pMinus;
    }

    public string Name
    {
        get
        {
            return Manufacturer + " - " + Model +
                " : " + Velocity.ToString() + "km/h";
        }
        set
        {
            Model = value;
        }
    }

    public void DrawObject()
    {
        Console.WriteLine
        (
            "      -----\n" +
            "_____/          \\_____ \n" +
            "|                      | \n" +
            "----(@)-----(@)--- \n"
        );
        Console.WriteLine(Name);
    }
}

```

7. Когда все классы и интерфейсы определены и реализованы можно их использовать. Продемонстрируем использование типов данных созданием соответствующих объектов в функции main.

```

static void Main(string[] args)
{
    Console.BackgroundColor = ConsoleColor.White;
    Console.ForegroundColor = ConsoleColor.Blue;
    Console.Clear();

    Console.Title = "Лабораторная работа №11";

    Human h = new Human("Иванов Иван Иванович", 50);
    h.Plus(5);
    h.Minus(1);
    h.DrawObject();

    Console.WriteLine("\n\n\n");

    Car car = new Car("Hyundai", "ix35", 120);
    car.Plus(25);
    car.Minus(11);
    car.DrawObject();

    Console.ReadKey();
}

```

8. Вид окна разработанного приложения представлен на рис. 9.1:

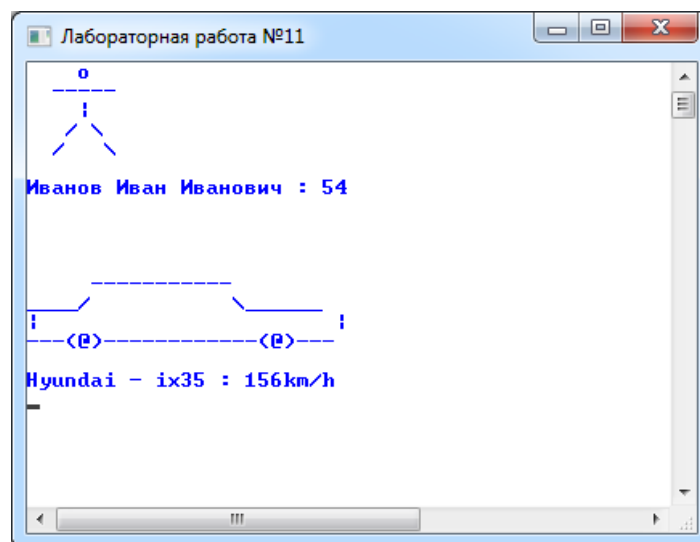


Рисунок 9.1 – Консольное приложение на основе интерфейсных типов

Диаграмма типов данных, созданная редактором Visual Studio, для разработанного приложения, показана на рис. 9.2:

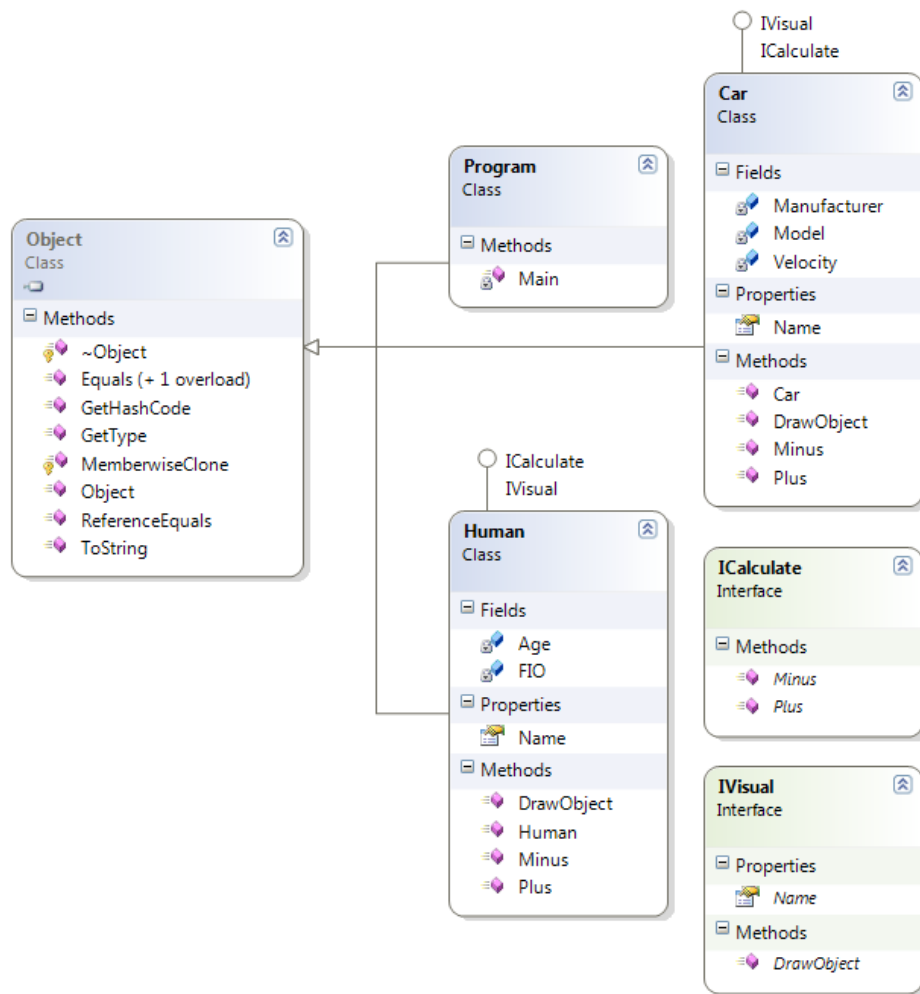


Рисунок 9.2 – Диаграмма классов приложения

9.3.2. Индивидуальное задание

Спроектируйте классы, наполните их требуемой функциональностью, определите интерфейс, продемонстрируйте использование объектов класса и методов интерфейса. Постройте диаграмму классов своего приложения средствами Visual Studio или с использованием сторонних редакторов UML-диаграмм.

Продемонстрируйте явление полиморфизма при использовании классов из индивидуального задания.

9.4. Вопросы для защиты работы

1. Чем отличается наследование интерфейсов от наследования реализации?
2. Поясните, каким образом проявляется полиморфное поведение объектов при реализации классами интерфейсов?
3. Сколько интерфейсов может реализовывать класс?
4. Как объявляются интерфейсные типы? Для чего используются интерфейсы?
5. Даны определения нескольких типов. Укажите ошибки (если есть) в представленном фрагменте:

```
public class MainObject: IString, IDraw
{
    int data;

    public string GetStr()
    {
        return "Данные: " + data.ToString();
    }

    public void Draw()
    {
        Console.WriteLine(data.ToString());
    }
}

public interface IDraw
{
    void Draw();
}

public interface IString
{
    string GetStr();
}
```

6. Даны определения нескольких типов. Укажите ошибки (если есть) в представленном фрагменте:

<pre>public interface IDraw { void Draw(); } public interface IString { string GetStr(); } public class Parent { protected int a; }</pre>	<pre>public class MainObject: IString, IDraw { int data; public string GetStr() { return "Данные: " + data.ToString(); } public void Draw() { Console.WriteLine(data.ToString()); } }</pre>
---	---

```
static void Main(string[] args)
{
    MainObject ob1 = new MainObject();
    ob1.Draw();

    Parent ob2 = new Parent();
    ob2.Draw();
}
```

ЛАБОРАТОРНАЯ РАБОТА 10. ОСНОВЫ РАБОТЫ С ФАЙЛАМИ

10.1. Цель и содержание

Цель лабораторной работы: научиться использовать механизмы файлового ввода-вывода.

Задачи лабораторной работы:

- научиться применять классы для работы с файлами;
- научиться применять классы для работы с каталогами;
- научиться использовать потоки ввода-вывода.

10.2. Теоретическая часть

10.2.1. Классы .NET Framework для реализации операций ввода-вывода

Весь ввод и вывод в .NET Framework подразумевает использование потоков. Поток – это абстрактное представление последовательного устройства. Последовательное устройство – это нечто такое, что хранит данные в линейной структуре и точно таким же образом обеспечивает доступ к ним: считывает или записывает по одному байту за одну единицу времени. Сохранение устройства абстрактным означает, что лежащие в основе источник/приемник данных могут быть скрыты. Такой уровень абстракции обеспечивает повторное использование кода и позволяет писать более обобщенные процедуры, потому что нет необходимости заботиться о действительной специфике передачи данных.

Для обработки файлов в C# необходима ссылка на пространство имен System.IO. При открытии файла создается объект, с которым ассоциируется поток. Потоки обеспечивают механизмы связи между файлами и программами.

Среда .NET Framework предоставляет все необходимые инструменты для эффективного использования файлов в приложениях.

Основные классы, необходимые программисту:

1. Object – исходный базовый класс для всех классов платформы .NET Framework и корень иерархии типов.

2. File – статический служебный класс, предоставляющий множество статических методов, которые дают возможность перемещать, создавать, копировать, удалять файлы, опрашивать и обновлять атрибуты, а также создавать объекты потоков FileStream.

3. Directory – статический служебный класс, предоставляющий множество статических методов для перемещения, копирования и удаления каталогов.

4. Path – служебный класс, используемый для манипулирования путевыми именами.

5. MarshalByRefObject – разрешает доступ к объектам через границы доменов приложения в приложениях, поддерживающих удаленное взаимодействие, это базовый класс для всех классов .NET, позволяющих удаленное взаимодействие.

6. FileInfo – представляет физический файл на диске, имеет методы для манипулирования этим файлом. Для любого объекта, который читает или пишет в этот файл, должен быть создан объект Stream. Все методы FileInfo доступны из объектной переменной, поэтому, если необходимо выполнить только одно действие, более эффективным может оказаться использование метода File, а не соответствующего экземпляра метода FileInfo. Для всех методов FileInfo требуется путь к файлу, с которым проводится операция. Все статические методы класса FileInfo выполняют проверку безопасности для всех методов. Если необходимо использовать объект неоднократно, рекомендуется использовать соответствующий метод экземпляра FileInfo, поскольку в этом случае проверка безопасности будет требоваться не всегда.

7. DirectoryInfo – представляет физический каталог на диске и предоставляет методы уровня экземпляра для манипулирования каталогом. Класс DirectoryInfo работает точно так же, как класс FileInfo. Это объект, представляющий отдельный каталог на машине. Подобно классу FileInfo, многие из вызовов методов дублируются между Directory и DirectoryInfo.

8. FileSystemInfo – служит базовым классом для FileInfo и DirectoryInfo, обеспечивая возможность работы с файлами и каталогами одновременно, используя полиморфизм.

9. Stream – предоставляет универсальное представление последовательности байтов. Класс Stream является абстрактным базовым классом всех потоков.

10. FileStream – представляет файл, который может быть записан, прочитан или то и другое.

11. TextReader – представляет средство чтения, позволяющее считывать последовательные наборы знаков. Этот класс является абстрактным базовым классом для StreamReader, который считывает символы из потоков.

12. TextWriter – представляет средство записи, позволяющее записывать последовательные наборы символов. Этот класс является абстрактным базовым классом для StreamWriter, который записывают символы в потоки.

13. StreamReader – читает символьные данные из потока и может быть создан с использованием класса FileStream в качестве базового.

14. StreamWriter – пишет символьные данные в поток и может быть создан с использованием класса FileStream в качестве базового.

15. Component – предоставляет базовую реализацию интерфейса IComponent и делает возможным совместное использование объектов разными приложениями.

16. FileSystemWatcher – используется для мониторинга файлов и каталогов и представляет события, которые приложение может перехватить, когда в этих объектах происходят какие-то изменения.

Таким образом, эта система классов включает в себя классы для работы с файлами (File, FileInfo), каталогами (Directory, DirectoryInfo) и потоками (FileStream, StreamReader, StreamWriter).

В большинстве случаев для разработки бизнес-приложений достаточно лишь четырех классов для манипулирования файловой системой. Эти классы расположены в пространстве имен System.IO и предназначены для работы с файловой системой компьютера, то есть для создания, удаления, переноса файлов и каталогов.

Первые два типа – Directory и File реализуют свои возможности с помощью статических методов, поэтому данные классы можно использовать без создания соответствующих объектов (экземпляров классов).

Следующие типы – DirectoryInfo и FileInfo обладают схожими функциональными возможностями с Directory и File, но порождены от класса FileSystemInfo, поэтому реализуются путем создания соответствующих экземпляров классов.

Класс FileSystemInfo предоставляет базовый функционал. Значительная часть членов FileSystemInfo предназначена для работы с общими характеристиками файла или каталога (метками времени, атрибутами и т. п.). Рассмотрим некоторые свойства FileSystemInfo (таблица 10.1).

Таблица 10.1 – Свойства класса FileSystemInfo

Свойство	Описание
Attributes	Позволяет получить или установить атрибуты для данного объекта файловой системы. Для этого свойства используются значения и перечисления FileAttributes
CreationTime	Позволяет получить или установить время создания объекта файловой системы
Exists	Может быть использовано для того, чтобы определить, существует ли данный объект файловой системы
Extension	Позволяет получить расширение для файла
FullName	Возвращает имя файла или каталога с указанием пути к нему в файловой системе
LastAccessTime	Позволяет получить или установить время последнего обращения к объекту файловой системы
LastWriteTime	Позволяет получить или установить время последнего внесения изменений в объект файловой системы
Name	Возвращает имя указанного файла. Это свойство доступно только для чтения. Для каталогов возвращает имя последнего каталога в иерархии, если это возможно. Если нет, возвращает полностью определенное имя

В FileSystemInfo предусмотрен набор методов. Например, метод Delete() – позволяет удалить объект файловой системы с жесткого диска, а Refresh() – обновить информацию об объекте файловой системы.

10.2.2. Классы для работы с каталогами файловой системы

Класс DirectoryInfo наследует члены класса FileSystemInfo и содержит дополнительный набор членов, которые предназначены для создания,

перемещения, удаления, получения информации о каталогах и подкаталогах в файловой системе. Наиболее важные члены класса содержатся в таблице 10.2.

Таблица 10.2 – Доступные члены класса DirectoryInfo

Член	Описание
Create() CreateSubDirectory()	Создают каталог (или подкаталог) по указанному пути в файловой системе
Delete()	Удаляет пустой каталог
GetDirectories()	Позволяет получить доступ к подкаталогам текущего каталога (в виде массива объектов DirectoryInfo)
GetFiles()	Позволяет получить доступ к файлам текущего каталога (в виде массива объектов FileInfo)
MoveTo()	Перемещает каталог и все его содержимое на новый адрес в файловой системе
Parent	Возвращает родительский каталог в иерархии файловой системы

Работа с типом DirectoryInfo начинается с того, что создается экземпляр класса (объект), при вызове конструктора в качестве параметра указывается путь к нужному каталогу. Если необходимо обратиться к текущему каталогу (то есть каталогу, в котором в настоящее время производится выполнение приложения), вместо параметра используется обозначение ".". Например:

```
// Создаем объект DirectoryInfo,  
// которому будет обращаться к текущему каталогу  
DirectoryInfo dir1 = new DirectoryInfo(".");  
  
// Создаем объект DirectoryInfo,  
// которому будет обращаться к каталогу d:\prim  
DirectoryInfo dir2 = new DirectoryInfo(@"d:\prim");
```

Если создается объект DirectoryInfo, который связывается с несуществующим каталогом, то будет сгенерировано исключение System.IO.DirectoryNotFoundException.

Свойство Attributes класса DirectoryInfo позволяет получить информацию об атрибутах объекта файловой системы. Возможные значения данного свойства приведены в следующей таблице 10.3.

Таблица 10.3 – Значения свойства Attributes

Значение	Описание
Archive	Этот атрибут используется приложениями при проведении резервного копирования, а в некоторых случаях - удаления старых файлов
Compressed	Определяет, что файл является сжатым
Directory	Определяет, что объект файловой системы является каталогом
Encrypted	Определяет, что файл является зашифрованным
Hidden	Определяет, что файл является скрытым (такой файл не будет выводиться при обычном просмотре каталога)
Normal	Определяет, что файл находится в обычном состоянии и для него установлены любые другие атрибуты. Этот атрибут не может использоваться с другими атрибутами
Offline	Файл (расположенный на сервере) кэширован в хранилище off-line на клиентском компьютере. Возможно, что данные этого файла уже устарели
Readonly	Файл доступен только для чтения
System	Файл является системным (то есть файл является частью операционной системы или используется исключительно операционной системой)

Через класс DirectoryInfo программист может собрать информацию о дочерних подкаталогах. Например:

```
static void printDirect(DirectoryInfo dir)
{
    Console.WriteLine("***** " + dir.Name + " *****");
    Console.WriteLine("FullName: {0}", dir.FullName);
    Console.WriteLine("Name: {0}", dir.Name);
    Console.WriteLine("Parent: {0}", dir.Parent);
    Console.WriteLine("Creation: {0}", dir.CreationTime);
    Console.WriteLine("Attributes: {0}", dir.Attributes.ToString());
    Console.WriteLine("Root: {0}", dir.Root);
}

static void Main(string[] args)
{
    DirectoryInfo dir = new DirectoryInfo(@"d:\prim");
    printDirect(dir);
    DirectoryInfo[] subDirects = dir.GetDirectories();
    Console.WriteLine("Найдено {0} подкаталогов", subDirects.Length);
    foreach (DirectoryInfo d in subDirects)
    {
        printDirect(d);
    }
}
```

Метод CreateSubdirectory() позволяет создать в выбранном каталоге как единственный подкаталог, так и множество подкаталогов (в том числе, и вложенных друг в друга). Например:

```
DirectoryInfo dir = new DirectoryInfo(@"d:\prim");

//создали подкаталог
dir.CreateSubdirectory("doc");

//создали вложенный подкаталог
dir.CreateSubdirectory(@"book\2008");
```

Метод MoveTo() позволяет переместить текущий каталог по заданному в качестве параметра адресу. При этом возможно произвести переименование каталога. Например:

```
DirectoryInfo dir = new DirectoryInfo(@"d:\prim\bmp");  
dir.MoveTo(@"d:\prim\letter\Николаев");
```

В данном случае каталог «bmp» перемещается по адресу «d:\prim\letter\Николаев». Так как имя перемещаемого каталога не совпадает с крайним правым именем в адресе нового местоположения каталога, то производится переименование.

Работать с каталогами файловой системы компьютера можно и при помощи класса Directory, функциональные возможности которого во многом совпадают с возможностями DirectoryInfo. Следует учитывать, что члены данного класса реализованы статически, поэтому для их использования нет необходимости создавать объект. Например:

```
// Создание подкаталога Новый  
Directory.CreateDirectory(@"d:\prim\Новый");  
  
// Перемещение каталога Новый в каталог 2012  
Directory.Move(@"d:\prim\Новый",  
              @"d:\prim\2012\Новый");  
  
//переименовали каталог Новый в Старый  
Directory.Move(@"d:\prim\2012\Новый",  
              @"d:\prim\2012\Старый");
```

Следует учитывать, что удаление каталога возможно только когда он пуст. На практике комбинируют использование классов Directory и DirectoryInfo.

10.2.3. Классы для работы с файлами

Класс FileInfo предназначен для организации доступа к физическому файлу, который содержится на жестком диске компьютера. Он позволяет получать информацию об этом файле (например, о времени его создания, размере, атрибутах), а также производить различные операции, например, по созданию файла или его удалению. Класс FileInfo наследует члены класса FileSystemInfo и содержит дополнительный набор членов, который приведен в следующей таблице 10.4

Таблица 10.4 – Члены класса FileInfo

Член	Описание
AppendText()	Создает объект StreamWriter для добавления текста к файлу.
CopyTo()	Копирует уже существующий файл в новый файл.
Create()	Создает новый файл и возвращает объект FileStream для взаимодействия с этим файлом.
CreateText()	Создает объект StreamWriter для записи текстовых данных в новый файл.
Delete()	Удаляет файл, которому соответствует объект FileInfo.
Directory	Возвращает каталог, в котором расположен данный файл.
DirectoryName	Возвращает полный путь к данному файлу в файловой системе.
Length	Возвращает размер файла.
MoveTo()	Перемещает файл в указанное пользователем место (этот метод позволяет одновременно переименовать данный файл).
Name	Позволяет получить имя файла.
Open()	Открывает файл с указанными пользователем правами доступа на чтение, запись или совместное использование с другими пользователями.
OpenRead()	Создает объект FileStream, доступный только для чтения.
OpenText()	Создает объект StreamReader (о нем также будет рассказано ниже), который позволяет считывать информацию из существующего текстового файла.
OpenWrite()	Создает объект FileStream, доступный для чтения и записи.

Большинство методов FileInfo возвращает объекты классов FileStream, StreamWriter, StreamReader и т. п., которые позволяют различным образом взаимодействовать с файлом, например, производить чтение или запись в него. Например:

```

// Создание объекта FileInfo
FileInfo f = new FileInfo("text.txt");

// Создание файла и потока
StreamWriter fOut =
    new StreamWriter(f.Create());

// Запись текста в файл
fOut.WriteLine("ОДИН ДВА ТРИ...");

// Закрытие файла
fOut.Close();

// Получение информации о файле
Console.WriteLine("Name: {0}", f.Name);
Console.WriteLine("File size: {0}",
    f.Length);
Console.WriteLine("Creation: {0}",
    f.CreationTime);
Console.WriteLine("Attributes: {0}",
    f.Attributes.ToString());

```

Доступ к физическим файлам можно получать и через статические методы класса File. Большинство методов объекта FileInfo представляют в этом смысле зеркальное отражение методов объекта File.

10.2.4. Потоки в системе ввода-вывода

Программы на языке C# 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, память) и записывать в различные источники (принтер, экран, файл, память).

Центральную часть потоковой системы C# занимает класс Stream пространства имен System.IO. Класс Stream представляет байтовый поток и является базовым для всех остальных потоковых классов. Производными от класса Stream являются классы потоков:

1. FileStream – байтовый поток, разработанный для файлового ввода-вывода.
2. BufferedStream – заключает в оболочку байтовый поток и добавляет буферизацию, которая во многих случаях увеличивает производительность программы.

3. `MemoryStream` – байтовый поток, который использует память для хранения данных.

Программист может реализовать собственные потоковые классы. Однако для подавляющего большинства приложений достаточно встроенных потоков.

10.2.5. Байтовый поток

Чтобы создать байтовый поток, связанный с файлом, создается объект класса `FileStream`. При этом в классе определено несколько конструкторов. Чаще всего используется конструктор, который открывает поток для чтения и (или) записи:

```
public FileStream(string path, FileMode mode);
```

Параметр `path` определяет имя файла, с которым будет связан поток ввода-вывода данных. Параметр `mode` определяет режим открытия файла, который может принимать одно из возможных значений, определенных перечислением `FileMode`:

- `FileMode.Append` – предназначен для добавления данных в конец файла;
- `FileMode.Create` – предназначен для создания нового файла, при этом если существует файл с таким же именем, то он будет предварительно удален;
- `FileMode.CreateNew` – предназначен для создания нового файла, при этом файл с таким же именем не должен существовать;
- `FileMode.Open` – предназначен для открытия существующего файла;
- `FileMode.OpenOrCreate` – если файл существует, то открывает его, в противном случае создает новый;
- `FileMode.Truncate` – открывает существующий файл, но усекает его длину до нуля.

Если попытка открыть файл оказалась неудачной, то генерируется одно из исключений:

- `FileNotFoundException` – файл невозможно открыть по причине его отсутствия;
- `IOException` – файл невозможно открыть из-за ошибки ввода-вывода;
- `ArgumentNullException` – имя файла представляет собой `null`-значение;
- `ArgumentException` – некорректен параметр `mode`;

- `SecurityException` – пользователь не обладает правами доступа;
- `DirectoryNotFoundException` – некорректно задан каталог.

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

```
public FileStream(string path, FileMode mode, FileAccess access);
```

Параметры `path` и `mode` имеют то же назначение, что и в предыдущей версии конструктора. Параметр `access`, 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.

После установления связи байтового потока с физическим файлом внутренний указатель потока устанавливается на начальный байт файла.

Для чтения очередного байта из потока, связанного с физическим файлом, используется метод `ReadByte()`. После прочтения очередного байта внутренний указатель перемещается на следующий байт файла. Если достигнут конец файла, то метод `ReadByte()` возвращает значение `-1`.

Для побайтовой записи данных в поток используется метод `WriteByte()`.

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.

```

static void Main(string[] args)
{
    try
    {
        // Создание потока для чтения из файла
        FileStream fileIn =
            new FileStream("ФайлСТекстом.txt",
                FileMode.Open,
                FileAccess.Read);
        // Создание потока для записи в файл
        FileStream fileOut =
            new FileStream("ФайлКопия.txt",
                FileMode.Create,
                FileAccess.Write);
        int i;

        // В цикле производится чтение одного файла
        // и одновременная запись содержимого
        // во второй файл
        while ((i = fileIn.ReadByte()) != -1)
        {
            // запись очередного байта в поток
            fileOut.WriteByte((byte)i);
        }

        // Закройте файлы
        fileIn.Close();
        fileOut.Close();
    }
    catch (Exception EX)
    {
        Console.WriteLine(EX.Message);
    }
}

```

В данном примере создаются два потока fileIn (для чтения байтов из файла в поток) и fileOut (для записи байтов из потока в файл). Каждый из потоков ассоциируется со своим файлом. Затем в цикле производится побайтовое чтение файла «ФайлСТекстом.txt» до момента, когда функция ReadByte() вернет значение -1 (то есть достигнут конец файла). При этом на каждой итерации цикла считывается один байт и на этой же итерации этот байт записывается в файл «ФайлКопия.txt». После всех операция оба потока закрываются. Весь код, осуществляющий работу с файлами заключен в конструкцию try ... catch. Это позволяет повысить устойчивость программы к ошибкам.

10.2.6. Символьный поток

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. В нем определено несколько конструкторов. Один из них записывается следующим образом:

```
public StreamWriter(Stream stream);
```

Параметр `stream` определяет имя уже открытого байтового потока. Этот конструктор генерирует исключение типа `ArgumentException`, если поток `stream` не открыт для вывода, и исключение типа `ArgumentNullException`, если он (поток) имеет `null`-значение.

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

```
public StreamWriter(string path);
```

Параметр `path` определяет имя открываемого файла.

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

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

И еще один вариант конструктора `StreamWriter`:

```
public StreamWriter(string path, bool append);
```

Параметр `path` определяет имя открываемого файла, а параметр `append` может принимать значение `true` – если нужно добавлять данные в конец файла, или `false` – если файл необходимо перезаписать.

Например:

```
// Добавляем символы в конец файла
StreamWriter fileOut_1 =
    new StreamWriter("МойФ.txt", true);
```

Объявив таким образом переменную fileOut_1 для записи данных в поток можно обратиться к методу WriteLine:

```
// Использование WriteLine
fileOut_1.WriteLine("Строка для записи");
```

В данном случае для записи используется метод, аналогичный статическому методу класса Console. Это действительно схожие механизмы ввода-вывода.

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

```
public StreamReader(Stream stream);
```

Параметр stream определяет имя уже открытого байтового потока.

Этот конструктор генерирует исключение типа ArgumentException, если поток stream не открыт для ввода. Создать экземпляр класса StreamReader 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:

```
public StreamReader(string path);
```

Параметр path определяет имя открываемого файла. Обратиться к данному конструктору можно следующим образом:

```
StreamReader fileIn =
    new StreamReader
        (@":c:\temp\Николаев.txt");
```

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

```
StreamReader fileIn =  
    new StreamReader(  
        @"c:\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. Тип `Encoding` реализован в пространстве имен `System.Text`.

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

Рассмотрим пример, в котором данные из одного файла копируются в другой, но уже с использованием классов `StreamWriter` и `StreamReader`.

```
static void Main(string[] args)  
{  
    // Создание символьных потоков  
    StreamReader fileIn =  
        new StreamReader(  
            "ФайлСТекстом.txt",  
            Encoding.GetEncoding(1251));  
    StreamWriter fileOut =  
        new StreamWriter(  
            "НовыйФайл.txt",  
            false);  
    string line;  
  
    // Построчное считывание  
    while ((line = fileIn.ReadLine()) != null)  
    {  
        // ... и запись строки  
        fileOut.WriteLine(line);  
    }  
  
    fileIn.Close();  
    fileOut.Close();  
}
```

В данном примере осуществляется копирование содержимого одного символьного файла в другой.

Таким образом, данный способ копирования одного файла в другой, дает тот же результат, что и при использовании байтовых потоков. Однако, его работа будет менее эффективной, т.к. будет тратиться дополнительное время на преобразование байтов в символы. У символьных потоков есть и свои преимущества. Например,

можно использовать регулярные выражения для поиска заданных фрагментов текста в файле.

10.2.7. Перенаправление стандартных потоков.

Тремя стандартными потоками, доступ к которым осуществляется через свойства `Console.Out`, `Console.In` и `Console.Error`, могут пользоваться все программы, работающие в пространстве имен `System`. Свойство `Console.Out` относится к стандартному выходному потоку. По умолчанию это консоль. Например, при вызове метода `Console.WriteLine()` информация автоматически передается в поток `Console.Out`. Свойство `Console.In` относится к стандартному входному потоку, источником которого по умолчанию является клавиатура. Например, при вводе данных с клавиатуры информация автоматически передается потоку `Console.In`, к которому можно обратиться с помощью метода `Console.ReadLine()`. Свойство `Console.Error` относится к ошибкам в стандартном потоке, источником которого также по умолчанию является консоль. Однако эти потоки могут быть перенаправлены на любое совместимое устройство ввода-вывода, например, на работу с физическими файлами.

Перенаправить стандартный поток можно с помощью методов `SetIn()`, `SetOut()` и `SetError()`, которые являются членами класса `Console`:

```
static void SetIn(TextReader input);  
static void SetOut(TextWriter output);  
static void SetError(TextWriter output);
```

Пример перенаправления потоков представлен в следующей программе, демонстрирующей, что стандартный поток вывода перенаправляется в один файл, а поток ввода – в другой:

```

static void Main(string[] args)
{
    int[,] MyArray;
    StreamReader file = new StreamReader("input.txt");
    // Перенаправление на file
    Console.SetIn(file);
    string line = Console.ReadLine();
    string[] mas = line.Split(' ');
    int n = int.Parse(mas[0]);
    int m = int.Parse(mas[1]);
    MyArray = new int[n, m];
    for (int i = 0; i < n; i++)
    {
        line = Console.ReadLine();
        mas = line.Split(' ');
        for (int j = 0; j < m; j++)
        {
            MyArray[i, j] = int.Parse(mas[j]);
        }
    }
    PrintArray("Исходный массив:", MyArray, n, m);
    file.Close();
}

static void PrintArray(string a, int[,] mas, int n, int m)
{
    // Перенаправление на file
    StreamWriter file=new StreamWriter("output.txt");
    Console.SetOut(file);
    Console.WriteLine(a);
    for (int i = 0; i < n; i++)
    {
        for (int j=0; j<m; j++) Console.Write("{0} ", mas[i,j]);
        Console.WriteLine();
    }
    file.Close();
}

```

10.3. Методика и порядок выполнения работы

10.3.1. Учебная задача

Для выполнения лабораторной работы необходимо спроектировать многомодульное приложение, использующее файлы для ввода и вывода информации.

Например, требуется разработать приложение, которое позволяет:

- генерировать матрицу случайных чисел с сохранением ее в файл;
- считывать матрицу из файла;

– выводить матрицу на экран.

Для выполнения данного задания-примера достаточно в отдельном проекте реализовать класс. Этот проект следует скомпилировать в виде dll-файла. Затем, в основной программе просто необходимо использовать данную библиотеку.

Класс библиотеки назовем Matrix. В нем определим следующие функции:

```
class Matrix
{
    private float[,] matrix;
    int m, n;

    // Конструктор
    public Matrix()...

    // Генерация матрицы заданного размера
    public void GenerateMatrix(int M, int N)...

    // Сохранение сгенерированной матрицы в файл
    public void SaveMatrix(string pFileName)...

    // Загрузка сохраненной матрицы из файла
    public Boolean LoadMatrix(string pFileName)...

    // Вывод матрицы в консоль
    public void PrintMatrix()...
}
```

Рисунок 10.1 – Основные методы класса Matrix

Листинг метода GenegateMatrix:

```
// Генерация матрицы заданного размера
public void GenerateMatrix(int M, int N)
{
    m = M; n = N;

    Random r = new Random(DateTime.Now.Millisecond);

    matrix = new float[M, N];

    for (int i = 0; i < M; i++)
        for (int j = 0; j < N; j++)
            matrix[i, j] = (float)r.Next(1000) / 973f;
}
```

Рисунок 10.2 – Метод для создания матрицы заданного размера и заполнения ее случайными числами

Листинг метода SaveMatrix:

```
// Сохранение сгенерированной матрицы в файл
public void SaveMatrix(string pFileName)
{
    if (matrix.Length > 0)
    {
        if (File.Exists(pFileName))
            File.Delete(pFileName);

        FileInfo f = new FileInfo(pFileName);
        TextWriter tw = f.CreateText();

        tw.WriteLine(m.ToString());
        tw.WriteLine(n.ToString());

        for (int i = 0; i < m; i++)
            for (int j = 0; j < n; j++)
                tw.WriteLine(i.ToString() + " " + j.ToString() + " " + matrix[i, j].ToString("E10"));

        tw.Close();
    }
}
```

Рисунок 10.3 – Метод для сохранения матрицы в файл

```
// Загрузка сохраненной матрицы из файла
public Boolean LoadMatrix(string pFileName)
{
    if (File.Exists(pFileName))
    {
        try
        {
            TextReader tr = File.OpenText(pFileName);
            m = Convert.ToInt32(tr.ReadLine());
            n = Convert.ToInt32(tr.ReadLine());

            matrix = new float[m, n];
            string line;
            string[] substring;

            for (int i = 0; i < m; i++)
                for (int j = 0; j < n; j++)
                {
                    line = tr.ReadLine();
                    substring = line.Split(new char[] { ' ' }, 3);
                    matrix[i, j] = Convert.ToSingle(substring[2]);
                }

            tr.Close();

            return true;
        }
        catch
        {
            return false;
        }
    }

    return false;
}
```

Рисунок 10.4 – Метод загрузки матрицы из файла

На рис. 10.4 представлен листинг метода LoadMatrix. Листинг метода PrintMatrix:

```
// Вывод матрицы в консоль
public void PrintMatrix()
{
    if (matrix.Length > 0)
    {
        for (int i = 0; i < m; i++)
        {
            for (int j = 0; j < n; j++)
                Console.Write(matrix[i, j].ToString("E3") + " ");
            Console.WriteLine();
        }
    }
}
```

Рисунок 10.5 – Метод для вывода матрицы

Основная программа для использования класса-матрицы представлена на рис. 10.6

```
class Program
{
    static void Main(string[] args)
    {
        Console.BackgroundColor = ConsoleColor.White;
        Console.ForegroundColor = ConsoleColor.Black;
        Console.Clear();

        Matrix m = new Matrix();

        m.GenerateMatrix(10, 5);
        m.SaveMatrix("FileForMatrix.txt");

        if (m.LoadMatrix("FileForMatrix.txt"))
            m.PrintMatrix();

        Console.ReadKey();
    }
}
```

Рисунок 10.6 – Работа с библиотечным классом

В результате работы данного кода будет создан файл FileForMatrix.txt следующего содержания:

```
1 10
2 5
3 0 0 1,1305241300E-001
4 0 1 9,2086333000E-001
5 0 2 3,6690646400E-001
6 0 3 8,7358683300E-002
7 0 4 8,3453238000E-001
8 1 0 8,6844807900E-001
9 1 1 4,7173690800E-001
10 1 2 4,7173690800E-001
11 1 3 9,8458373500E-001
12 1 4 4,9331963100E-001
13 2 0 4,8098662500E-001
14 2 1 1,5005138500E-001
15 2 2 6,9886946700E-001
16 2 3 1,9218911200E-001
17 2 4 0,0733500000E-001
```

Рисунок 10.7 – Результирующий файл

На экран будет выведена следующая информация:

```
1,131E-001 9,209E-001 3,669E-001 8,736E-002 8,345E-001
8,684E-001 4,717E-001 4,717E-001 9,846E-001 4,933E-001
4,810E-001 1,501E-001 6,989E-001 1,922E-001 8,972E-001
7,790E-001 9,373E-001 3,443E-001 1,014E+000 1,912E-001
6,249E-001 1,706E-001 9,003E-001 2,888E-001 8,088E-001
1,010E+000 9,054E-001 9,322E-001 3,535E-001 3,350E-001
5,385E-001 3,279E-001 7,698E-001 7,040E-001 8,633E-001
6,886E-002 3,422E-001 6,177E-001 9,424E-001 2,806E-001
1,922E-001 7,194E-003 7,883E-001 1,439E-002 7,893E-001
1,552E-001 8,068E-001 2,734E-001 1,418E-001 8,191E-001
```

Рисунок 10.8 – Вывод программы в консоль

10.3.2. Индивидуальное задание

Выполните индивидуальное задание, согласно предложенному варианту. В каждом варианте необходимо спроектировать многомодульное приложение (оптимально 2 модуля). Исходные данные в каждом варианте программа получает из входного файла.

Вариант	Задание
1	Программа рассчитывает произведение двух матриц, которые хранятся в разных файлах.
2	Программа рассчитывает сумму двух матриц, которые хранятся в разных файлах.
3	Программа находит максимальный элемент в двух матрицах, которые хранятся в разных файлах.
4	Программа рассчитывает сумму диагональных элементов двух матриц, которые хранятся в разных файлах.
5	Программа рассчитывает сумму элементов с четной суммой индексов в двух матрицах, которые хранятся в разных файлах.
6	Программа рассчитывает разность сумм элементов матриц, которые хранятся в разных файлах.
7	Программа рассчитывает сумму элементов главной диагонали первой матрицы и сумму элементов второстепенной диагонали второй матрицы. Матрицы хранятся в разных файлах.
8	Программа рассчитывает сумму элементов четных столбцов в двух матрицах, которые хранятся в разных файлах.
9	Программа рассчитывает сумму диагональных элементов четных столбцов в двух матрицах, которые хранятся в разных файлах.
10	Программа рассчитывает сумму элементов четных строк в двух матрицах, которые хранятся в разных файлах.
11	Программа рассчитывает сумму элементов, находящихся в четном столбце и нечетной строке, в двух матрицах, которые хранятся в разных файлах.
12	Программа рассчитывает сумму элементов четных строк и расположенных на второстепенной диагонали в обеих матрицах, которые хранятся в разных файлах.
13	Программа рассчитывает произведение элементов в двух матрицах, которые хранятся в разных файлах.
14	Программа рассчитывает сумму первой матрицы и матрицы транспонированной относительно второй. Обе матрицы хранятся в отдельных файлах.
15	Программа рассчитывает произведение элементов диагонали первой матрицы и сумму всех элементов второй матрицы. Обе матрицы хранятся в отдельных файлах.
16	Программа рассчитывает сумму элементов четных строк в двух матрицах, которые хранятся в разных файлах.
17	Программа находит максимальный элемент в двух матрицах, которые хранятся в разных файлах.
18	Программа рассчитывает сумму элементов четных строк и расположенных на второстепенной диагонали в обеих матрицах, которые хранятся в разных файлах.

Вариант	Задание
19	Программа рассчитывает сумму первой матрицы и матрицы транспонированной относительно второй. Обе матрицы хранятся в отдельных файлах.
20	Программа рассчитывает произведение двух матриц, которые хранятся в разных файлах.
21	Программа рассчитывает сумму элементов четных столбцов в двух матрицах, которые хранятся в разных файлах.
22	Программа рассчитывает сумму диагональных элементов четных столбцов в двух матрицах, которые хранятся в разных файлах.
23	Программа рассчитывает произведение элементов диагонали первой матрицы и сумму всех элементов второй матрицы. Обе матрицы хранятся в отдельных файлах.
24	Программа рассчитывает сумму элементов с четной суммой индексов в двух матрицах, которые хранятся в разных файлах.
25	Программа рассчитывает разность сумм элементов матриц, которые хранятся в разных файлах.
26	Программа рассчитывает произведение двух матриц, которые хранятся в разных файлах. Для каждой матрицы (двух исходных и результирующей) выводится среднее арифметическое ее элементов.
27	Программа рассчитывает сумму и разность двух матриц, которые хранятся в разных файлах.
28	Программа находит минимальный, максимальный элементы в двух матрицах, которые хранятся в разных файлах.
29	Программа рассчитывает сумму диагональных элементов двух матриц, которые хранятся в разных файлах.
30	Программа рассчитывает сумму элементов с четной суммой индексов в двух матрицах, которые хранятся в разных файлах.
31	Программа рассчитывает разность сумм элементов матриц, которые хранятся в разных файлах.
32	Программа рассчитывает сумму элементов главной диагонали первой матрицы и сумму элементов второстепенной диагонали второй матрицы. Матрицы хранятся в разных файлах.
33	Программа рассчитывает сумму элементов четных столбцов в двух матрицах, которые хранятся в разных файлах.
34	Программа рассчитывает сумму диагональных элементов четных столбцов в двух матрицах, которые хранятся в разных файлах.
35	Программа рассчитывает сумму элементов четных строк в двух матрицах, которые хранятся в разных файлах.
36	Программа рассчитывает сумму элементов, находящихся в четном столбце и нечетной строке, в двух матрицах, которые хранятся в разных файлах.

Вариант	Задание
37	Программа рассчитывает сумму элементов четных строк и расположенных на второстепенной диагонали в обеих матрицах, которые хранятся в разных файлах.
38	Программа рассчитывает произведение элементов в двух матрицах, которые хранятся в разных файлах.
39	Программа рассчитывает сумму первой матрицы и матрицы транспонированной относительно второй. Обе матрицы хранятся в отдельных файлах.
40	Программа рассчитывает произведение элементов диагонали первой матрицы и сумму всех элементов второй матрицы. Обе матрицы хранятся в отдельных файлах.

10.4. Контрольные вопросы

1. Какие классы для работы с файловой системой вы знаете?
2. Что такое сборка?
3. Как определить проект по умолчанию в решении Visual Studio?
4. Какие классы отвечают за представление файлов в программе?
5. Что такое поток? Какие типы классов потоков используются при работе с файлами?
6. Опишите последовательность действий при необходимости записать одну строку в файл. Приведите примеры использования различных классов.
7. Перечислите классы для работы с каталогами.
8. Поясните принцип работы синтаксической конструкции **try ... catch**.

ЛАБОРАТОРНАЯ РАБОТА 11. ОБРАБОТКА ДАННЫХ В ФАЙЛАХ

11.1. Цель и содержание

Цель лабораторной работы: научиться использовать механизмы файлового ввода-вывода.

Задачи лабораторной работы:

- научиться применять классы для работы с файлами;
- научиться применять классы для работы с каталогами;
- научиться использовать потоки ввода-вывода.

11.2. Теоретическая часть

Повторить теоретический материал лабораторной работы 10. При выполнении данной лабораторной работы рекомендуется использовать символьный поток.

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. В нем определено несколько конструкторов. Один из них записывается следующим образом:

```
public StreamWriter(Stream stream);
```

Параметр `stream` определяет имя уже открытого байтового потока. Этот конструктор генерирует исключение типа `ArgumentException`, если поток `stream` не открыт для вывода, и исключение типа `ArgumentNullException`, если он (поток) имеет `null`-значение.

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

```
public StreamWriter(string path);
```

Параметр path определяет имя открываемого файла.

Например, создать экземпляр класса StreamWriter можно следующим образом:

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

И еще один вариант конструктора StreamWriter:

```
public StreamWriter(string path, bool append);
```

Параметр path определяет имя открываемого файла, а параметр append может принимать значение true – если нужно добавлять данные в конец файла, или false – если файл необходимо перезаписать.

Например:

```
// Добавляем символы в конец файла  
StreamWriter fileOut_1 =  
    new StreamWriter("МойФ.txt", true);
```

Объявив таким образом переменную fileOut_1 для записи данных в поток можно обратиться к методу WriteLine:

```
// Использование WriteLine  
fileOut_1.WriteLine("Строка для записи");
```

В данном случае для записи используется метод, аналогичный статическому методу класса Console. Это действительно схожие механизмы ввода-вывода.

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

```
public StreamReader(Stream stream);
```

Параметр stream определяет имя уже открытого байтового потока.

Этот конструктор генерирует исключение типа ArgumentException, если поток stream не открыт для ввода. Создать экземпляр класса StreamReader 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:

```
public StreamReader(string path);
```

Параметр `path` определяет имя открываемого файла. Обратиться к данному конструктору можно следующим образом:

```
StreamReader fileIn =  
    new StreamReader  
        (@":c:\temp\Николаев.txt");
```

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

```
StreamReader fileIn =  
    new StreamReader(  
        @":c:\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`. Тип `Encoding` реализован в пространстве имен `System.Text`.

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

Рассмотрим пример, в котором данные из одного файла копируются в другой, но уже с использованием классов `StreamWriter` и `StreamReader`.

```

static void Main(string[] args)
{
    // Создание символьных потоков
    StreamReader fileIn =
        new StreamReader(
            "ФайлСТекстом.txt",
            Encoding.GetEncoding(1251));
    StreamWriter fileOut =
        new StreamWriter(
            "НовыйФайл.txt",
            false);
    string line;

    // Построчное считывание
    while ((line = fileIn.ReadLine()) != null)
    {
        // ... и запись строки
        fileOut.WriteLine(line);
    }

    fileIn.Close();
    fileOut.Close();
}

```

В данном примере осуществляется копирование содержимого одного символьного файла в другой.

Таким образом, данный способ копирования одного файла в другой, дает тот же результат, что и при использовании байтовых потоков. Однако, его работа будет менее эффективной, т.к. будет тратиться дополнительное время на преобразование байтов в символы. У символьных потоков есть и свои преимущества. Например, можно использовать регулярные выражения для поиска заданных фрагментов текста в файле.

11.3. Методика и порядок выполнения работы

11.3.1. Учебная задача

Условие. На рисунке 11.1 представлен фрагмент файла `data_example.csv`, который содержит данные о пользователях в следующем формате:

ID, Fname, Lname, Email, Gender, Salary, HasChildren

где **ID** – идентификатор, **FName** – имя, **Lname** – фамилия, **Email** – адрес электронной почты, **Gender** – пол, **Salary** – зарплата, **HasChildren** – есть дети или нет. Все поля разделены запятыми, данные каждого пользователя начинаются с

новой строки. Пропусков в данных нет. Файл содержит около 1000 строк. Первая строка – заголовок. Необходимо ответить на следующие вопросы:

- 1) определить количество мужчин и женщин;
- 2) найти самую высокую зарплату среди женщин и среди мужчин (вывести этих пользователей);
- 3) посчитать количество пользователей с детьми и без;
- 4) посчитать суммарный фонд заработной платы для четырех категорий, образованных признаками Gender и HasChildren.

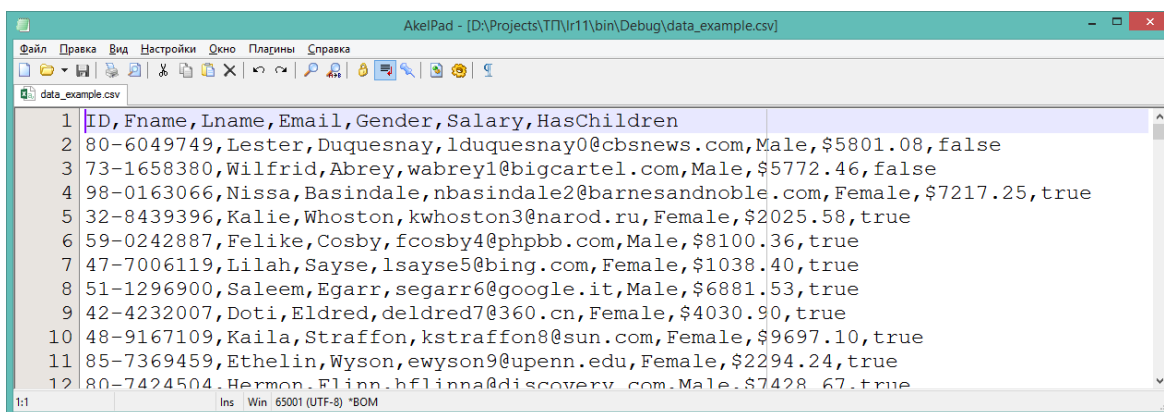


Рисунок 11.1 – Фрагмент исходного файла

Решение. Для отработки алгоритмов необходимо создать копию исходного файла данных, которая будет содержать 10-20 строк. На этом файле следует проводить разработку и тестирование приложения.

Спроектируем класс Person для представления одной строки файла. Проект разработанного класса показан на рисунке 11.2. Для представления поля Gender создадим тип-перечисление GenderType, для конвертации текстовой строки из файла в объект типа Person объявим метод Create(). Объявление основных типов реализовано в файле Person.cs (рисунок 11.3)

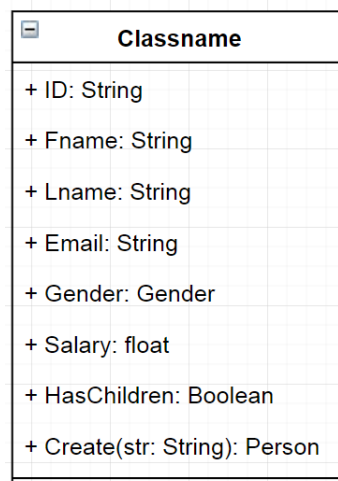


Рисунок 11.2 – Диаграмма UML для класса Person

```

1  using System;
2
3  namespace lr11
4  {
5      enum GenderType
6      {
7          Male,
8          Female
9      }
10
11     class Person
12     {
13         String ID { get; set; }
14         String Fname { get; set; }
15         String Lname { get; set; }
16         String Email { get; set; }
17         public GenderType Gender { get; set; }
18         public float Salary { get; set; }
19         public Boolean HasChildren { get; set; }
20
21         Person Create(String str)
22         {
23             Person p = new Person();
24
25             return p;
26         }
27     }
28 }

```

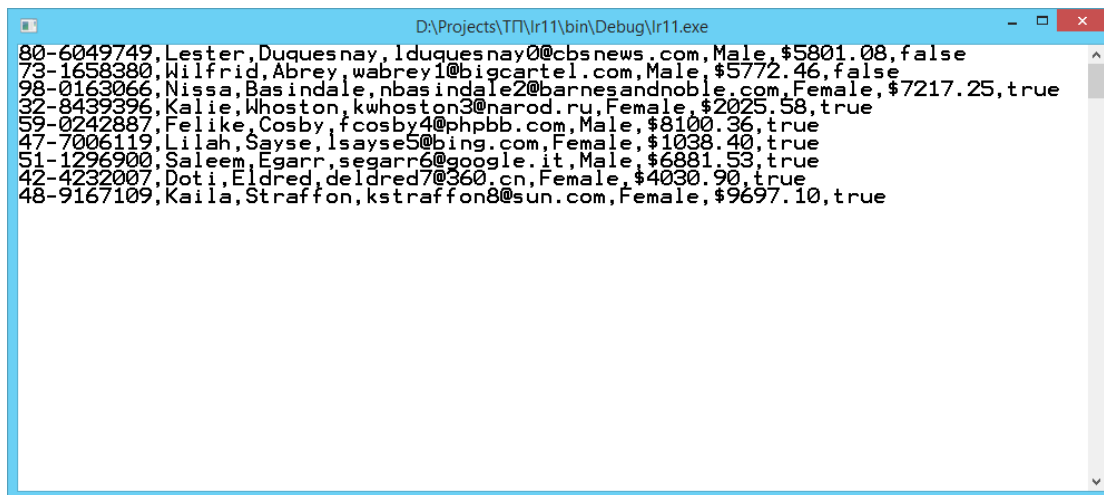
Рисунок 11.3 – Листинг Person.cs

Теперь реализуем набросок главной программы (рисунок 11.4).

```
1  using System;
2  using System.Collections.Generic;
3  using System.Linq;
4  using System.IO;
5
6  namespace lr11
7  {
8      class Program
9      {
10         static void Main(string[] args)
11         {
12             StreamReader f_in = new StreamReader("data_example_small.csv");
13             #if !DEBUG
14                 TextWriter save_out = Console.Out;
15                 var new_out = new StreamWriter(@"lr11_output.txt");
16                 Console.SetOut(new_out);
17             #endif
18
19             List<Person> all = new List<Person>();
20             String line = f_in.ReadLine();
21             while((line = f_in.ReadLine()) != null)
22             {
23                 Console.WriteLine(line);
24             }
25
26             #if !DEBUG
27                 Console.SetOut(save_out);
28                 new_out.Close();
29             #else
30                 Console.ReadKey();
31             #endif
32         }
33     }
34 }
```

Рисунок 11.4 – Листинг модуля Program.cs с функцией Main()

Данный набросок программы выводит все считанный из файла строки на консоль (в режиме DEBUG). На рисунке 11.5 показан результат работы приложения в режиме DEBUG.



```
D:\Projects\TPI\lr11\bin\Debug\lr11.exe
80-6049749,Lester,Duquesnay,lduquesnay0@cbsnews.com,Male,$5801.08,false
73-1658380,Wilfrid,Abrey,wabrey1@bigcartel.com,Male,$5772.46,false
98-0163066,Nissa,Basindale,nbasindale2@barnesandnoble.com,Female,$7217.25,true
32-8439396,Kalie,Whoston,kwhoston3@narod.ru,Female,$2025.58,true
59-0242887,Felike,Cosby,fcosby4@phpbb.com,Male,$8100.36,true
47-7006119,Lilah,Sayse,lsayse5@bing.com,Female,$1038.40,true
51-1296900,Saleem,Egarr,segarr6@google.it,Male,$6881.53,true
42-4232007,Doti,Eldred,deldred7@360.cn,Female,$4030.90,true
48-9167109,Kaila,Straffon,kstraffon8@sun.com,Female,$9697.10,true
```

Рисунок 11.4 – Листинг модуля Program.cs с функцией Main()

Теперь можно доработать метод `Person.Create()` для конвертации строки входного файла в объект типа `Person` (рисунок 11.5). В этом же листинге добавлены вспомогательные методы для удобства представления класса `Person`.

```
11  class Person
12  {
13      String ID { get; set; }
14      String Fname { get; set; }
15      String Lname { get; set; }
16      String Email { get; set; }
17      public GenderType Gender { get; set; }
18      public float Salary { get; set; }
19      public Boolean HasChildren { get; set; }
20
21      public static Person Create(String str)
22      {
23          Person p = new Person();
24          string[] e = str.Split(',');
25          p.ID = e[0].Trim();
26          p.Fname = e[1].Trim();
27          p.Lname = e[2].Trim();
28          p.Email = e[3].Trim();
29          String tmp = e[4].Trim();
30          if (tmp == "Male")
31              p.Gender = GenderType.Male;
32          else p.Gender = GenderType.Female;
33          p.Salary = Convert.ToSingle(e[5].TrimStart('$').Replace('.', ','));
34          p.HasChildren = Convert.ToBoolean(e[6].Trim());
```

```

35
36         return p;
37     }
38
39     public override string ToString()
40     {
41         String s = string.Format(
42             "*****\n" +
43             "ID: {0}, Имя: {1}, Фамилия: {2}, ({3})\n" +
44             "E-mail: {4}, З/п: {5}, Дети: {6}",
45             ID, Fname, Lname, GenderToStr(Gender),
46             Email, Salary, HasChildrenToStr(HasChildren));
47
48         return s;
49     }
50
51     private static String GenderToStr(GenderType g)
52     {
53         if (g == GenderType.Male) return "М";
54         else return "Ж";
55     }
56
57     private static String HasChildrenToStr(Boolean g)
58     {
59         if (g == true) return "Есть";
60         else return "Нет";
61     }
62 }
63

```

Рисунок 11.4 – Доработка класса Person

Модифицируем метод Main(), теперь он позволяют заполнить список объектов типа Person (рисунок 11.5) и вывести список объектов в консоль в режиме debug. В листинге представлен фрагмент, отражающий работу с коллекцией объектов Person. Работа программы на данном этапе разработке представлена на рисунке 11.6.

После подготовительных действий можно перейти к решению заданий учебной задачи.

1. Найдем количество пользователей мужского и женского пола. Для этого в метод Main() добавим следующий код (рисунок 11.7)

```

18 List<Person> all = new List<Person>();
19
20 try
21 {
22
23     String line = f_in.ReadLine();
24     while((line = f_in.ReadLine()) != null)
25     {
26         all.Add(Person.Create(line));
27     }
28 } catch (Exception ex)
29 {
30     Console.WriteLine(ex.Message);
31 }
32
33 Console.WriteLine("Всего пользователей: {0}", all.Count);
34 foreach (var p in all)
35     Console.WriteLine(p);

```

Рисунок 11.5 – Доработка метода Main()

```

*****
ID: 73-1658380, Имя: Wilfrid, Фамилия: Abrey, (М)
E-mail: wabrey1@bigcartel.com, З/п: 5772,46, Дети: Нет
*****
ID: 98-0163066, Имя: Nissa, Фамилия: Basindale, (Ж)
E-mail: nbasindale2@barnesandnoble.com, З/п: 7217,25, Дети: Есть
*****
ID: 32-8439396, Имя: Kalie, Фамилия: Whoston, (Ж)
E-mail: kwhoston3@narod.ru, З/п: 2025,58, Дети: Есть
*****
ID: 59-0242887, Имя: Felike, Фамилия: Cosby, (М)
E-mail: fcosby4@phpbb.com, З/п: 8100,36, Дети: Есть
*****
ID: 47-7006119, Имя: Lilah, Фамилия: Sayse, (Ж)
E-mail: lsayse5@bing.com, З/п: 1038,4, Дети: Есть
*****
ID: 51-1296900, Имя: Saleem, Фамилия: Egarr, (М)
E-mail: segarr6@google.it, З/п: 6881,53, Дети: Есть
*****
ID: 42-4232007, Имя: Doti, Фамилия: Eldred, (Ж)
E-mail: deldred7@360.cn, З/п: 4030,9, Дети: Есть
*****
ID: 48-9167109, Имя: Kaila, Фамилия: Straffon, (Ж)
E-mail: kstraffon8@sun.com, З/п: 9697,1, Дети: Есть
*****

```

Рисунок 11.6 – Демонстрация работы программы на этапе разработанного функционала класса Person

```

Console.WriteLine("***** Задача 1 *****");
int mCount = all.FindAll(p => p.Gender == GenderType.Male).ToList().Count;
Console.WriteLine("Количество мужчин: {0}", mCount);
Console.WriteLine("Количество женщин: {0}", all.Count - mCount);

```

Рисунок 11.7 – Вычисление количества пользователей мужского и женского пола

Таким образом, первая задача решена в 3 строки. Переходим к решению второй задачи.

2. Требуется найти самую высокую зарплату среди женщин и среди мужчин (вывести этих пользователей). Для нахождения этих величин используем код, представленный на рисунке 11.8.

```
Console.WriteLine("***** Задача 2 *****");
float male_max = (from p in all
                  where p.Gender == GenderType.Male
                  select p.Salary).Max();
Person rich_man = (from p in all
                  where (p.Gender == GenderType.Male) &&
                        (p.Salary == male_max)
                  select p).First();
Console.WriteLine(rich_man);

float female_max = (from p in all
                   where p.Gender == GenderType.Female
                   select p.Salary).Max();
Person rich_woman = (from p in all
                   where (p.Gender == GenderType.Female) &&
                         (p.Salary == female_max)
                   select p).First();
Console.WriteLine(rich_woman);
```

Рисунок 11.8 – Вычисление максимальной зарплаты для женщин и мужчин

Для решения задач используется язык LINQ, который является встроенным «подъязыком» технологии .NET. Можно решить задачи классическими алгоритмами с использованием циклов. Переходим к заданию 3.

3. Требуется посчитать количество пользователей с детьми и без. Листинг для данной задачи представлен на рисунке 11.9. В данном случае решаем классическим перебором всей коллекции объектов Person.

```
Console.WriteLine("\n***** Задача 3 *****");
int Count_Yes = 0, Count_No = 0;
for (int i = 0; i < all.Count; i++)
{
    if (all[i].HasChildren) Count_Yes++;
    else Count_No++;
}
Console.WriteLine("С детьми: {0}, без детей {1}",
                  Count_Yes, Count_No);
```

Рисунок 11.8 – Вычисление количества пользователей с детьми и без

4. Решим последнюю задачу. Требуется посчитать суммарный фонд заработной платы для четырех категорий, образованных признаками **Gender** и **HasChildren**. То есть нужно разбить всех пользователей на 4 группы (рисунок 11.9) и в каждой посчитать суммарный фонд заработной платы. В данной задаче дополнительно выведем суммарный фонд заработной платы по всем пользователям. Решение представлено на рисунке 11.10.

	Есть дети	Нет детей
Мужчины		
Женщины		

Рисунок 11.9 – Группы, на которые разбивает всех пользователей признаки **Gender** и **HasChildren**

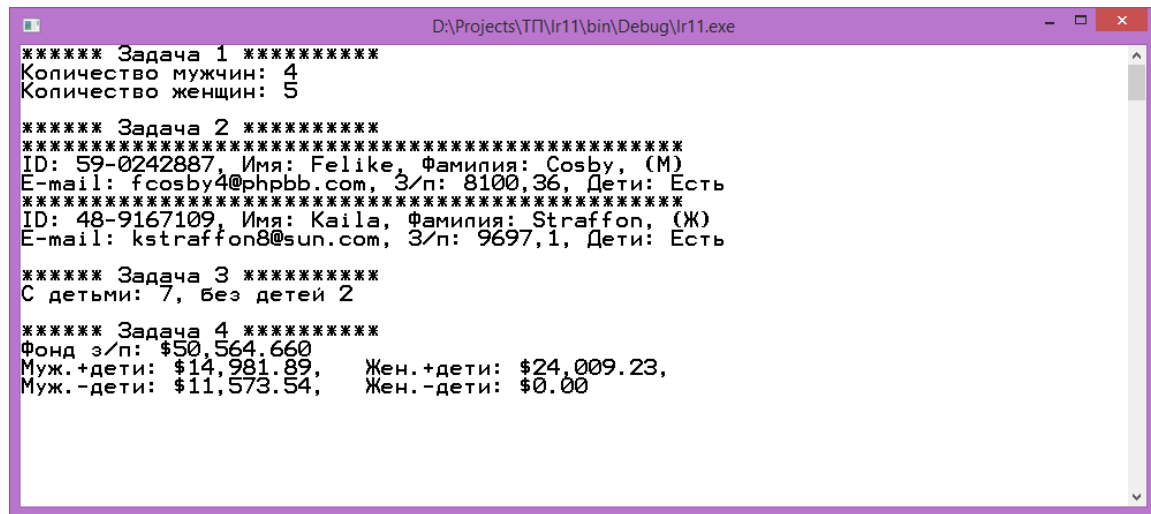
```

Console.WriteLine("\n***** Задача 4 *****");
float salary_total = (from p in all
                      select p.Salary).Sum();
float m_haschildren = (from p in all
                      where (p.Gender == GenderType.Male) && p.HasChildren
                      select p.Salary).Sum();
float f_haschildren = (from p in all
                      where (p.Gender == GenderType.Female) && p.HasChildren
                      select p.Salary).Sum();
float m_nochildren = (from p in all
                      where (p.Gender == GenderType.Male) && !p.HasChildren
                      select p.Salary).Sum();
float f_nochildren = (from p in all
                      where (p.Gender == GenderType.Female) && !p.HasChildren
                      select p.Salary).Sum();
var f = System.Globalization.CultureInfo.GetCultureInfo("en-us");
Console.WriteLine("Фонд з/п: {0}\n" +
    "Муж.+дети: {1},\t Жен.+дети: {2},\n" +
    "Муж.-дети: {3},\t Жен.-дети: {4}",
    salary_total.ToString("C3", f),
    m_haschildren.ToString("C", f), f_haschildren.ToString("C", f),
    m_nochildren.ToString("C", f), f_nochildren.ToString("C", f));

```

Рисунок 11.10 – Расчет фондов заработной платы

В результате всех доработок debug-программа выводит в консоль следующие данные, которые представляют собой решение подзадач в учебной задаче:



```
***** Задача 1 *****
Количество мужчин: 4
Количество женщин: 5

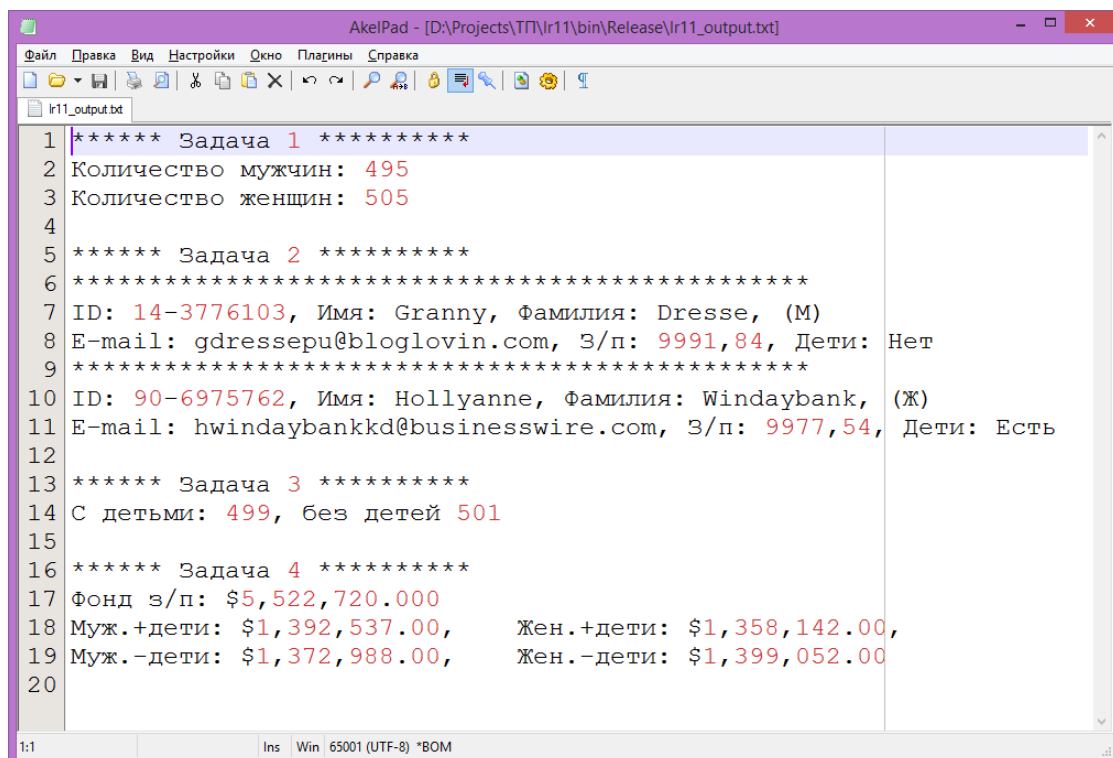
***** Задача 2 *****
ID: 59-0242887, Имя: Felike, Фамилия: Cosby, (М)
E-mail: fcosby4@phpbb.com, З/п: 8100,36, Дети: Есть
ID: 48-9167109, Имя: Kaila, Фамилия: Straffon, (Ж)
E-mail: kstraffon8@sun.com, З/п: 9697,1, Дети: Есть

***** Задача 3 *****
С детьми: 7, Без детей 2

***** Задача 4 *****
Фонд з/п: $50,564.660
Муж.+дети: $14,981.89, Жен.+дети: $24,009.23,
Муж.-дети: $11,573.54, Жен.-дети: $0.00
```

Рисунок 11.11 – Вывод приложения в режиме debug

После разработки debug-версии приложения можно перейти к сборке release-сборки и замене ограниченного, тестового набора данных на полный набор данных, содержащий большое количество записей. На рисунке 11.12 представлен выходной файл lr11_output.txt, содержащий вывод release-сборки.



```
1 ***** Задача 1 *****
2 Количество мужчин: 495
3 Количество женщин: 505
4
5 ***** Задача 2 *****
6 *****
7 ID: 14-3776103, Имя: Granny, Фамилия: Dresse, (М)
8 E-mail: gdressepu@bloglovin.com, З/п: 9991,84, Дети: Нет
9 *****
10 ID: 90-6975762, Имя: Hollyanne, Фамилия: Windaybank, (Ж)
11 E-mail: hwindaybankkd@businesswire.com, З/п: 9977,54, Дети: Есть
12
13 ***** Задача 3 *****
14 С детьми: 499, без детей 501
15
16 ***** Задача 4 *****
17 Фонд з/п: $5,522,720.000
18 Муж.+дети: $1,392,537.00, Жен.+дети: $1,358,142.00,
19 Муж.-дети: $1,372,988.00, Жен.-дети: $1,399,052.00
20
```

Рисунок 11.12 – Выходной файл приложения в режиме release

11.3.2. Индивидуальное задание

В таблице представлены задания, которые необходимо выполнить с использованием языка программирования. В каждом варианте необходимо использовать свой набор данных (файлы с данными необходимо получить у ведущего преподавателя).

Вариант	Задание
1	1. Количество сотрудников в каждой из 3-х категорий; 2. Суммарный фонд з/п и сумму по каждой категории; 3. Количество сотрудников с дипломом; 4. Максимальную зарплату среди сотрудников без диплома.
2	1. Количество продуктов в каждой из 3-х категорий; 2. Суммарный объем поставок в валюте по каждой категории; 3. Суммарный объем в валюте всей выборки; 4. Количество просроченных продуктов.
3	1. Самый дорогой продукт для каждого из 4-х поставщиков; 2. Товар с минимальным количеством в каждой категории; 3. Объем просрочки в валюте; 4. Среднюю цену у каждого поставщика.
4	1. Количество сотрудников в каждой из 4-х компаний; 2. Среднюю з/п в каждой из 4-х компаний; 3. Количество сотрудников, владеющих английским языком; 4. Количество сотрудников, получающих более \$3000,00.
5	1. Количество билетов с пунктом отправления из Ставрополя; 2. Средняя цена билета для каждого из четырех пунктов отправления; 3. Количество билетов, купленных женщинами и мужчинами; 4. Количество билетов с одинаковыми пунктами отправления и прибытия.
6	Дан список целей, которые описываются параметрами. 1. Определите среднее значение Param1 по каждой из категорий; 2. Определите сумму параметров Param2 для каждой категории; 3. Сколько целей снабжены оружием, а сколько – нет; 4. Найти объект с максимальным значением Param1.
7	Дан список сотрудников. 1. Количество сотрудников без логина; 2. Средняя заработная плата по каждому из 4-х отделов; 3. Максимальная з/п сотрудников у которых есть собака или кошка; 4. Средняя з/п у сотрудников с оружием и без.

8	Дан список доменов. Требуется определить: 1. Количество доменов для которых не указан domain или user; 2. Средняя цена домена по каждой из 4-х категорий; 3. Максимальная цена домена среди продающихся и не продающихся; 4. Суммарная стоимость доменов без доменного имени.
9	В файле содержатся сведения о сотрудниках компании, их зарплатах и сумме налога с зарплаты. 1. У какого сотрудника минимальная з/п? Какой сотрудник получает минимальную сумму после уплаты налогов? 2. У скольких сотрудников не указан навык (skill)? 3. Определите среднюю з/п для сотрудников с e-mail и без. 4. Определите сумму налога по всей выборке.
10	В файле содержатся данные о счетах клиентов банка, которые отражают приход и расход по счету, а также сумму налога, которую следует оплатить. 1. Сколько клиентов имеют отрицательный баланс счета, т.е. приход меньше расхода? 2. Вычислите среднюю сумму налогов по счетам. 3. Найдите клиента с максимальными расходами. 4. Вычислите средний доход по счету для клиентов без e-mail.
11	В файле содержатся сведения о товарах на складе (ID, наименование, поставщик, цена, количество, категория, скидка в %). 1. Определите количество товаров со скидкой более 25%. 2. Вычислите суммарный объем (в валюте) товаров, для которых не указал поставщик. 3. Для каждого из 4-х поставщиков определите среднюю цену товаров. 4. Найдите минимальную цену товара с учетом скидки.
12	В файле содержатся сведения о производстве товаров работниками (ID, сотрудник, категория товара, з/п рабочего, количество произведенных товаров, цена за единицу товара). 1. Определите сколько рабочих получают больше, чем вырабатывают продукции. 2. Суммарный объем произведенной продукции (в валюте) без категории. 3. Суммарный объем в валюте произведенной продукции по каждой из 4-х категорий. 4. Самый эффективный сотрудник: с максимальной разницей между произведенным продуктом и зарплатой.

13	В файле содержатся сведения о сотрудниках компании, их зарплатах и сумме налога с зарплаты. 1. Максимальная и минимальная з/п. Вывести соответствующих сотрудников. 2. Средняя з/п сотрудников без указанного навыка (skill)? 3. Количество сотрудников без e-mail. 4. Суммарный налог.
14	В файле содержатся данные о товарах (ID, наименование, категория (4 шт.), количество, цена, признак просрочки, признак наличия скидки). 1. Средняя цена товаров для просроченных и непросроченных товаров. 2. Суммарный объем в валюте товаров без категории. 3. Самый дорогой товар в каждой категории. 4. Количество наименований товаров без скидки.
15	В файле содержатся данные о счетах клиентов банка, которые отражают доход (plus) и расход (minus) по счету, а также сумму налога, которую следует оплатить (tax). 1. Суммарный расход по счетам клиентов, у которых не указан e-mail. 2. Клиент с самым большим балансом с учетом уплаты налогов. 3. Средний доход по счетам с положительным балансом. 4. Суммарная сумма налогов у клиентов с отрицательным балансом.
16	1. Сумма фонда з/п в каждой из 3-х категорий сотрудников; 2. Количество сотрудников с дипломом и без; 3. Количество сотрудников с з/п менее \$50000,00; 4. Максимальную зарплату среди сотрудников без диплома и минимальную у сотрудников с дипломом.
17	1. Средняя цена в каждой из 3-х категорий; 2. Товар с максимальной ценой; 3. Суммарный объем в валюте непросроченных продуктов; 4. Количество просроченных продуктов.
18	В файле содержатся сведения о товарах на складе (ID, наименование, поставщик, цена, количество, категория, скидка в %). 1. Определите среднюю цену с учетом скидки по каждому из 4-х поставщиков. 2. Найдите самый дорогой и самый дешевый товары. 3. По каждой из 4-х категорий найдите товар с минимальным остатком. 4. Количество товаров без указания поставщика.
19	Дан список сотрудников. 1. Сколько сотрудников не имеют логина или пароля; 2. Средняя з/п сотрудников с семьей и без; 3. Сколько сотрудников имеют машину, но не имеют семьи; 4. Минимальная и максимальная з/п в каждом из 4-х отделов.

20	1. Суммарный объем в валюте для каждого из 4-х поставщиков; 2. Количество наименований товаров дешевле \$15,00; 3. Суммарный объем просрочки в валюте у каждого поставщика; 4. Количество товаров с ценой более \$50,00.
21	В файле содержатся данные о счетах клиентов банка, которые отражают приход и расход по счету, а также сумму налога, которую следует оплатить. 1. Вычислите средний баланс по счетам с положительным балансом. 2. Количество счетов с отрицательным балансом. 3. Сколько клиентов имеют баланс, недостаточный для уплаты налога? 4. Найдите клиента с максимальным балансом счета.
22	1. Максимальную з/п в каждой из 4-х компаний; 2. Количество сотрудников в каждой из 4-х компаний; 3. Количество сотрудников в каждой языковой группе; 4. Количество сотрудников, получающих зарплату выше средней по всей выборке.
23	В файле содержатся сведения о производстве товаров работниками (ID, сотрудник, категория товара, з/п рабочего, количество произведенных товаров, цена за единицу товара). 1. Определите сколько рабочих получают меньше, чем вырабатывают продукции. 2. Количество единиц произведенной продукции без категории. 3. Суммарный объем в валюте произведенной продукции по каждой из 4-х категорий. 4. Самый неэффективный сотрудник: с минимальной разницей между произведенным продуктом и зарплатой.
24	В файле содержатся данные о товарах (ID, наименование, категория (4 шт.), количество, цена, признак просрочки, признак наличия скидки). 1. Максимальная цена товара для просроченных и непросроченных товаров. 2. Количество товаров без категории. 3. Самый дорогой товар с учетом скидки в каждой категории. 4. Объем товаров в валюте с учетом скидки.
25	В файле содержатся сведения о сотрудниках компании, их зарплатах и сумме налога с зарплаты. 1. Максимальная и минимальная з/п после уплаты налога. Вывести соответствующих сотрудников. 2. Средняя з/п сотрудников без указанного навыка (skill). 3. Количество сотрудников с e-mail. 4. Объем фонда заработной платы.

26	В файле содержатся сведения о товарах на складе (ID, наименование, поставщик, цена, количество, категория, скидка в %). 1. Определите количество наименований товаров в каждой из 4-х категорий. 2. Вычислите суммарный объем (в валюте) товаров по каждому из 4-х поставщиков, а также по товарам, у которых не указан поставщик. 3. Найдите самый дорогой товар с учетом скидки. 4. Найдите товар с минимальным количеством.
27	В файле содержатся сведения о производстве товаров работниками (ID, сотрудник, категория товара, з/п рабочего, количество произведенных товаров, цена за единицу товара). 1. Определите сколько рабочих получают меньше, чем вырабатывают продукции. 2. Количество единиц произведенной продукции без категории. 3. Суммарный объем в валюте произведенной продукции по каждой из 4-х категорий. 4. Количество сотрудников, получающих более 50% от суммы производимого продукта.
28	1. Максимальную з/п среди англоговорящих сотрудников; 2. Средняя з/п в каждой из 4-х компаний; 3. Количество сотрудников в каждой языковой группе; 4. Сотрудник с максимальной з/п.
29	В файле содержатся данные о счетах клиентов банка, которые отражают приход (plus) и расход (minus) по счету, а также сумму налога, которую следует оплатить (tax). 1. Максимальный расход по всем клиентам. 2. Количество счетов с положительным балансом. 3. Суммарный положительный и отрицательный балансы. 4. Средний баланс для клиентов с отрицательным балансом.
30	В файле содержатся данные о товарах (ID, наименование, категория (4 шт.), количество, цена, признак просрочки, признак наличия скидки). 1. Максимальная цена товара для просроченных и непросроченных товаров. 2. Суммарный объем в валюте товаров без категории с учетом скидки. 3. Самый дорогой товар с учетом скидки в каждой категории. 4. Количество наименований товаров без скидки.

31	В файле содержатся сведения о товарах на складе (ID, наименование, поставщик, цена, количество, категория, скидка в %). 1. Определите количество наименований товаров в каждой из 4-х категорий. 2. Вычислите суммарный объем (в валюте) по товарам, у которых не указан поставщик. 3. Найдите самый дорогой товар в каждой категории. 4. Найдите товар с максимальной скидкой.
32	1. Средняя цена для каждого из 4-х поставщиков; 2. Суммарный объем товаров от каждого поставщика; 3. Количество просрочки от каждого поставщика; 4. Количество товаров с ценой более \$50,00.
33	В файле содержатся данные о счетах клиентов банка, которые отражают приход (plus) и расход (minus) по счету, а также сумму налога, которую следует оплатить (tax). 1. Максимальный доход по всем клиентам. 2. Количество счетов с отрицательным балансом. 3. Суммарный положительный и отрицательный балансы. 4. Средний баланс для клиентов с отрицательным балансом.
34	Дан список целей, которые описываются параметрами. 1. Определите максимальное значение Param1 по каждой из категорий; 2. Определите сумму параметров Param3 для каждой категории 3. Сколько целей снабжены оружием, а сколько – нет; 4. Найти объект с максимальным значением Param2.
35	В файле содержатся сведения о товарах на складе (ID, наименование, поставщик, цена, количество, категория, скидка в %). 1. Определите среднюю цену товаров со скидкой более 25%. 2. Вычислите максимальную цену среди товаров, для которых не указа поставщик. Выведите этот товар. 3. Для каждого из 4-х поставщиков определите товар с максимальной ценой. 4. Найдите товар с максимальной ценой с учетом скидки и без учета скидки.
36	1. Средняя цена билета с отправлением из Ставрополя; 2. Количество билетов для каждого их четырех пунктов отправления; 3. Суммарный объем проданных билетов, оформленных на женщин и мужчин; 4. Максимальная и минимальная цена билета в Москву.

37	В файле содержатся данные о товарах (ID, наименование, категория (4 шт.), количество, цена, признак просрочки, признак наличия скидки). 1. Минимальная цена товара для просроченных и непросроченных товаров. 2. Суммарный объем в валюте товаров без категории с учетом скидки. 3. Самый дорогой товар с учетом скидки в каждой категории. 4. Средняя цена товаров в каждой категории.
38	В файле содержатся данные о счетах клиентов банка, которые отражают доход (plus) и расход (minus) по счету, а также сумму налога, которую следует оплатить (tax). 1. Количество клиентов с отрицательным балансом. 2. Клиент с самым большим балансом с учетом уплаты налогов. 3. Средний доход по счетам с отрицательным балансом. 4. Суммарная сумма налогов.
39	Даны сведения о товарах на складах компании (ID, наименование товара, категория товара, количество товара, цена за единицу, склад размещения). 1. Объем товаров в валюте по каждому из складов, а также для товаров без склада. 2. Максимальную цену по каждой категории. 3. Средняя цена товаров, для которых не указана категория. 4. Самый дешевый товар.
40	Дан список доменов. Требуется определить: 1. Суммарная стоимость доменов для которых не указан domain или user; 2. максимальная цена домена по каждой из 4-х категорий; 3. Средняя цена домена среди продающихся и не продающихся; 4. Количество доменов без доменного имени.

11.4. Контрольные вопросы

1. Дайте определение понятия «файл».
2. Опишите формат файла *.csv.
3. Поясните назначение различных конфигураций (debug или release) при сборке приложения в среде Visual Studio.
4. Приведите пример кода, реализующего различное поведение программы при использовании различных режимов сборки (debug или release).

5. Что такое поток? Какие типы классов потоков используются при работе с файлами?

6. Опишите последовательность действий при необходимости записать одну строку в файл. Приведите примеры использования различных классов.

СПИСОК ЛИТЕРАТУРЫ

Основная литература

1. Павлова, Е. А. Технологии разработки современных информационных систем на платформе Microsoft .NET : учебное пособие / Павлова Е. А. - Москва : Интернет-Университет Информационных Технологий (ИНТУИТ), 2016. - 128 с. - Книга находится в базовой версии ЭБС IPRbooks. - ISBN 978-5-9963-0003-7.

2. Пугачев, С. В. Разработка приложений для Windows 8 на языке C# / С. Пугачев, А. Шериев, К. Кичинский. - СПб. : БХВ-Петербург, 2013. - 416 с. : ил. - (Профессиональное программирование). - ISBN 978-56-9775-0846-9.

Дополнительная литература

1. Зиборов, В. В. Visual C# 2012 на примерах / Виктор Зиборов. - Санкт-Петербург : БХВ-Петербург, 2013. - 473 с. : ил. ; 24 см. - ISBN 978-5-9775-0888-9.

2. Прайс, Д. Visual C#2.0.NET. Полное руководство : [учеб. пособие] : пер. с англ. / Дж. Прайс, М. Гандрэрлой. - СПб. : КОРОНА-век, 2011. - 736 с. : ил. - Прил.: с. 673. - ISBN 978-5-7931-0555-2.

3. Чеповский, А. Common Intermediate Language и системное программирование в Microsoft .NET / А. Чеповский ; А. Макаров ; С. Скоробогатов. - 2-е изд., исправ. - Москва : Национальный Открытый Университет «ИНТУИТ», 2016. - 399 с. - (Основы информатики и математики). - ISBN 5-94774-410-4.

Перечень Интернет-ресурсов

1. <http://metanit.com> - сайт посвящен различным языкам и технологиям программирования, компьютерам, мобильным платформам и ИТ-технологиям.

2. <https://msdn.microsoft.com/magazine/> - Интернет-журнал о технологиях разработки Microsoft.

3. <https://professorweb.ru> - информационный ресурс, посвященный разработке приложений на основе технологии .NET.

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РФ
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ
УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«СЕВЕРО-КАВКАЗСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

Методические указания

по организации и проведению самостоятельной работы
по дисциплине

«Объектно-ориентированное программирование»

для студентов направления подготовки

09.03.02 «Информационные системы и технологии»

направленность (профиль)

**«Прикладное программирование в интеллектуальных информационных
системах»**

Ставрополь, 2026

1. Общие положения

Самостоятельная работа – планируемая учебная, учебно-исследовательская, научно-исследовательская работа студентов, выполняемая во внеаудиторное (аудиторное) время по заданию и при методическом руководстве преподавателя, но без его непосредственного участия (при частичном непосредственном участии преподавателя, оставляющем ведущую роль за работой студентов).

Самостоятельная работа студентов (СРС) в ВУЗе является важным видом учебной и научной деятельности студента. Самостоятельная работа студентов играет значительную роль в рейтинговой технологии обучения.

К основным видам самостоятельной работы студентов относятся:

- формирование и усвоение содержания конспекта лекций на базе рекомендованной лектором учебной литературы, включая информационные образовательные ресурсы (электронные учебники, электронные библиотеки и др.);
- написание докладов;
- подготовка к семинарам, практическим и лабораторным работам, их оформление;
- составление аннотированного списка статей из соответствующих журналов по отраслям знаний (педагогических, психологических, методических и др.);
- выполнение учебно-исследовательских работ, проектная деятельность;
- подготовка практических разработок и рекомендаций по решению проблемной ситуации;
- выполнение домашних заданий в виде решения отдельных задач, проведения типовых расчетов, расчетно-компьютерных и индивидуальных работ по отдельным разделам содержания дисциплин и т.д.;
- компьютерный текущий самоконтроль и контроль успеваемости на базе электронных обучающих и аттестующих тестов;
- выполнение курсовых работ (проектов) в рамках дисциплин;
- выполнение выпускной квалификационной работы и др.

Методика организации самостоятельной работы студентов зависит от структуры, характера и особенностей изучаемой дисциплины, объема часов на ее изучение, вида заданий для самостоятельной работы студентов, индивидуальных качеств студентов и условий учебной деятельности.

Процесс организации самостоятельной работы студентов включает в себя следующие этапы:

- подготовительный (определение целей, составление программы, подготовка методического обеспечения, подготовка оборудования);
- основной (реализация программы, использование приемов поиска информации, усвоения, переработки, применения, передачи знаний, фиксирование результатов, самоорганизация процесса работы);
- заключительный (оценка значимости и анализ результатов, их систематизация, оценка эффективности программы и приемов работы, выводы о направлениях оптимизации труда).

Самостоятельная работа по дисциплине «Объектно-ориентированное программирование» направлена на формирование следующих **компетенций**:

Код, формулировка компетенции	Код, формулировка индикатора
ПК-1	ПК-1 _{ид-1.1} – Демонстрирует знания в области применения алгоритмов и их адаптации к практическим задачам
	ПК-1 _{ид-1.2} – Разрабатывает приложения на языке высокого уровня в том числе с использованием интеллектуальных технологий

	ПК-1 _{ИД-1.3} – Применяет современные автоматизированные технологии тестирования и отладки программного обеспечения
	ПК-1 _{ИД-1.4} – Применяет современные технологии непрерывного развертывания и доставки программного обеспечения
	ПК-1 _{ИД-1.5} – Разрабатывает приложения для различных целевых платформ (мобильные, серверные, встраиваемые, оконные)

2. Цель и задачи самостоятельной работы

Ведущая цель организации и осуществления СРС совпадает с целью обучения студента – формирование набора общенаучных, профессиональных и специальных компетенций будущего бакалавра по соответствующему направлению подготовки

При организации СРС важным и необходимым условием становятся формирование умения самостоятельной работы для приобретения знаний, навыков и возможности организации учебной и научной деятельности. Целью самостоятельной работы студентов является овладение фундаментальными знаниями, профессиональными умениями и навыками деятельности по профилю, опытом творческой, исследовательской деятельности. Самостоятельная работа студентов способствует развитию самостоятельности, ответственности и организованности, творческого подхода к решению проблем учебного и профессионального уровня.

Задачами СРС являются:

- систематизация и закрепление полученных теоретических знаний и практических умений студентов;
- углубление и расширение теоретических знаний;
- формирование умений использовать нормативную, правовую, справочную документацию и специальную литературу;
- развитие познавательных способностей и активности студентов: творческой инициативы, самостоятельности, ответственности и организованности;
- формирование самостоятельности мышления, способностей к саморазвитию, самосовершенствованию и самореализации;
- развитие исследовательских умений;
- использование материала, собранного и полученного в ходе самостоятельных занятий на семинарах, на практических и лабораторных занятиях, при написании курсовых и выпускной квалификационной работ, для эффективной подготовки к итоговым зачетам и экзаменам.

3. Технологическая карта самостоятельной работы студента

Коды реализуемых компетенций	Вид деятельности студентов	Средства и технологии оценки	Объем часов, в том числе (акад.)		
			СРС	Контактная работа с преподавателем	Всего
5 семестр					
ПК-1	Самостоятельное изучение литературы и источников	Собеседование	8	2	10
ПК-1	Подготовка лабораторным занятиям	Защита ЛР	34 / 18	10	44
ПК-1	Выполнение курсовой работы	Защита КР	30 / 18	6	36
Итого за 5 семестр			72 / 36	18	90 / 36
Итого			72 / 36	18	90 / 36

4. Порядок выполнения самостоятельной работы студентом

4.1. Методические рекомендации по работе с учебной литературой

При работе с книгой необходимо подобрать литературу, научиться правильно ее читать, вести записи. Для подбора литературы в библиотеке используются алфавитный и систематический каталоги.

Важно помнить, что рациональные навыки работы с книгой - это всегда большая экономия времени и сил.

Правильный подбор учебников рекомендуется преподавателем, читающим лекционный курс. Необходимая литература может быть также указана в методических разработках по данному курсу.

Изучая материал по учебнику, следует переходить к следующему вопросу только после правильного уяснения предыдущего, описывая на бумаге все выкладки и вычисления (в том числе те, которые в учебнике опущены или на лекции даны для самостоятельного вывода).

При изучении любой дисциплины большую и важную роль играет самостоятельная индивидуальная работа.

Особое внимание следует обратить на определение основных понятий курса. Студент должен подробно разбирать примеры, которые поясняют такие определения, и уметь строить аналогичные примеры самостоятельно. Нужно добиваться точного представления о том, что изучаешь. Полезно составлять опорные конспекты. При изучении материала по учебнику полезно в тетради (на специально отведенных полях) дополнять конспект лекций. Там же следует отмечать вопросы, выделенные студентом для консультации с преподавателем.

Выводы, полученные в результате изучения, рекомендуется в конспекте выделять, чтобы они при перечитывании записей лучше запоминались.

Опыт показывает, что многим студентам помогает составление листа опорных сигналов, содержащего важнейшие и наиболее часто употребляемые формулы и понятия. Такой лист помогает запомнить формулы, основные положения лекции, а также может служить постоянным справочником для студента.

Чтение научного текста является частью познавательной деятельности. Ее цель – извлечение из текста необходимой информации. От того насколько осознанна читающим собственная внутренняя установка при обращении к печатному слову (найти нужные сведения, усвоить информацию полностью или частично, критически проанализировать материал и т.п.) во многом зависит эффективность осуществляемого действия.

Выделяют **четыре основные установки в чтении научного текста:**

информационно-поисковый (задача – найти, выделить искомую информацию)

усваивающая (усилия читателя направлены на то, чтобы как можно полнее осознать и запомнить как сами сведения излагаемые автором, так и всю логику его рассуждений)

аналитико-критическая (читатель стремится критически осмыслить материал, проанализировав его, определив свое отношение к нему)

творческая (создает у читателя готовность в том или ином виде – как отправной пункт для своих рассуждений, как образ для действия по аналогии и т.п. – использовать суждения автора, ход его мыслей, результат наблюдения, разработанную методику, дополнить их, подвергнуть новой проверке).

Основные виды систематизированной записи прочитанного:

Аннотирование – предельно краткое связное описание просмотренной или прочитанной книги (статьи), ее содержания, источников, характера и назначения;

Планирование – краткая логическая организация текста, раскрывающая содержание и структуру изучаемого материала;

Тезирование – лаконичное воспроизведение основных утверждений автора без привлечения фактического материала;

Цитирование – дословное выписывание из текста выдержек, извлечений, наиболее существенно отражающих ту или иную мысль автора;

Конспектирование – краткое и последовательное изложение содержания прочитанного.

Конспект – сложный способ изложения содержания книги или статьи в логической последовательности. Конспект аккумулирует в себе предыдущие виды записи, позволяет всесторонне охватить содержание книги, статьи. Поэтому умение составлять план, тезисы, делать выписки и другие записи определяет и технологию составления конспекта.

Методические рекомендации по составлению конспекта:

1. Внимательно прочитайте текст. Уточните в справочной литературе непонятные слова. При записи не забудьте вынести справочные данные на поля конспекта;
2. Выделите главное, составьте план;
3. Кратко сформулируйте основные положения текста, отметьте аргументацию автора;
4. Законспектируйте материал, четко следуя пунктам плана. При конспектировании старайтесь выразить мысль своими словами. Записи следует вести четко, ясно.
5. Грамотно записывайте цитаты. Цитируя, учитывайте лаконичность, значимость мысли.

В тексте конспекта желательно приводить не только тезисные положения, но и их доказательства. При оформлении конспекта необходимо стремиться к емкости каждого предложения. Мысли автора книги следует излагать кратко, заботясь о стиле и выразительности написанного. Число дополнительных элементов конспекта должно быть логически обоснованным, записи должны распределяться в определенной последовательности, отвечающей логической структуре произведения. Для уточнения и дополнения необходимо оставлять поля.

Овладение навыками конспектирования требует от студента целеустремленности, повседневной самостоятельной работы.

4.2. Методические рекомендации по подготовке к практическим и лабораторным занятиям

Для того чтобы практические и лабораторные занятия приносили максимальную пользу, необходимо помнить, что упражнение и решение задач проводятся по вычитанному на лекциях материалу и связаны, как правило, с детальным разбором отдельных вопросов лекционного

курса. Следует подчеркнуть, что только после усвоения лекционного материала с определенной точки зрения (а именно с той, с которой он излагается на лекциях) он будет закрепляться на практических занятиях как в результате обсуждения и анализа лекционного материала, так и с помощью решения проблемных ситуаций, задач. При этих условиях студент не только хорошо усвоит материал, но и научится применять его на практике, а также получит дополнительный стимул (и это очень важно) для активной проработки лекции.

При самостоятельном решении задач нужно обосновывать каждый этап решения, исходя из теоретических положений курса. Если студент видит несколько путей решения проблемы (задачи), то нужно сравнить их и выбрать самый рациональный. Полезно до начала вычислений составить краткий план решения проблемы (задачи). Решение проблемных задач или примеров следует излагать подробно, вычисления располагать в строгом порядке, отделяя вспомогательные вычисления от основных. Решения при необходимости нужно сопровождать комментариями, схемами, чертежами и рисунками.

Следует помнить, что решение каждой учебной задачи должно доводиться до окончательного логического ответа, которого требует условие, и по возможности с выводом. Полученный ответ следует проверить способами, вытекающими из существа данной задачи. Полезно также (если возможно) решать несколькими способами и сравнить полученные результаты. Решение задач данного типа нужно продолжать до приобретения твердых навыков в их решении.

4.3. Методические рекомендации по самопроверке знаний

После изучения определенной темы по записям в конспекте и учебнику, а также решения достаточного количества соответствующих задач на практических занятиях и самостоятельно студенту рекомендуется, провести самопроверку усвоенных знаний, ответив на контрольные вопросы по изученной теме.

В случае необходимости нужно еще раз внимательно разобраться в материале.

Иногда недостаточность усвоения того или иного вопроса выясняется только при изучении дальнейшего материала. В этом случае надо вернуться назад и повторить плохо усвоенный материал. Важный критерий усвоения теоретического материала - умение решать задачи или пройти тестирование по пройденному материалу. Однако следует помнить, что правильное решение задачи может получиться в результате применения механически заученных формул без понимания сущности теоретических положений.

4.4. Методические рекомендации по написанию научных текстов (докладов, докладов, эссе, научных статей и т.д.)

Перед тем, как приступить к написанию научного текста, важно разобраться, какова истинная цель вашего научного текста - это поможет вам разумно распределить свои силы и время.

Во-первых, сначала нужно определиться с идеей научного текста, а для этого необходимо научиться либо относиться к разным явлениям и фактам несколько критически (своя идея – как иная точка зрения), либо научиться увлекаться какими-то известными идеями, которые нуждаются в доработке (идея – как оптимистическая позиция и направленность на дальнейшее совершенствование уже известного). Во-вторых, научиться организовывать свое время, ведь, как известно, свободное (от всяких глупостей) время – важнейшее условие настоящего творчества, для него наконец-то появляется время. Иногда именно на организацию такого времени уходит немалая часть сил и талантов.

Писать следует ясно и понятно, стараясь основные положения формулировать четко и недвусмысленно (чтобы и самому понятно было), а также стремясь структурировать свой текст. Каждый раз надо представлять, что ваш текст будет кто-то читать и ему захочется сориентироваться в нем, быстро находить ответы на интересующие вопросы (заодно представьте себя на месте такого человека). Понятно, что работа, написанная «сплошным текстом» (без

заголовков, без выделения крупным шрифтом наиболее важным мест и т. п.), у культурного читателя должна вызывать брезгливость и даже жалость к автору (исключения составляют некоторые древние тексты, когда и жанр был иной и к текстам относились иначе, да и самих текстов было гораздо меньше – не то, что в эпоху «информационного взрыва» и соответствующего «информационного мусора»).

Объем текста и различные оформительские требования во многом зависят от принятых в конкретном учебном заведении порядков.

Доклад – это самостоятельное исследование студентом определенной проблемы, комплекса взаимосвязанных вопросов.

Доклад не должна составляться из фрагментов статей, монографий, пособий. Кроме простого изложения фактов и цитат, в доклад е должно проявляться авторское видение проблемы и ее решения.

Рассмотрим основные этапы подготовки
а студентом.

Выполнение доклада начинается с выбора темы.

Затем студент приходит на первую консультацию к руководителю, которая предусматривает:

- обсуждение цели и задач работы, основных моментов избранной темы;
- консультирование по вопросам подбора литературы;
- составление предварительного плана.

Следующим этапом является работа с литературой. Необходимая литература подбирается студентом самостоятельно.

После подбора литературы целесообразно сделать рабочий вариант плана работы. В нем нужно выделить основные вопросы темы и параграфы, раскрывающие их содержание.

Составленный список литературы и предварительный вариант плана уточняются, согласуются на очередной консультации с руководителем.

Затем начинается следующий этап работы – изучение литературы. Только внимательно читая и конспектируя литературу, можно разобраться в основных вопросах темы и подготовиться к самостоятельному (авторскому) изложению содержания доклада. Конспектируя первоисточники, необходимо отразить основную идею автора и его позицию по исследуемому вопросу, выявить проблемы и наметить задачи для дальнейшего изучения данных проблем.

Систематизация и анализ изученной литературы по проблеме исследования позволяют студенту написать работу.

Рабочий вариант текста доклада предоставляется руководителю на проверку. На основе рабочего варианта текста руководитель вместе со студентом обсуждает возможности доработки текста, его оформление. После доработки доклад сдается на кафедру для его оценивания руководителем.

Требования к написанию доклада

Написание 1 доклада является обязательным условием выполнения плана СРС по любой дисциплине профессионального цикла.

Тема доклада может быть выбрана студентом из предложенных в рабочей программе или фонде оценочных средств дисциплины, либо определена самостоятельно, исходя из интересов студента (в рамках изучаемой дисциплины). Выбранную тему необходимо согласовать с преподавателем.

Доклад должен быть написан научным языком.

Объем доклада должен составлять 20-25 стр.

Структура доклада:

- Введение (не более 3-4 страниц). Во введении необходимо обосновать выбор темы, ее актуальность, очертить область исследования, объект исследования, основные цели и задачи исследования.

- Основная часть состоит из 2-3 разделов. В них раскрывается суть исследуемой проблемы, проводится обзор мировой литературы и источников Интернет по предмету

исследования, в котором дается характеристика степени разработанности проблемы и авторская аналитическая оценка основных теоретических подходов к ее решению. Изложение материала не должно ограничиваться лишь описательным подходом к раскрытию выбранной темы. Оно также должно содержать собственное видение рассматриваемой проблемы и изложение собственной точки зрения на возможные пути ее решения.

- Заключение (1-2 страницы). В заключении кратко излагаются достигнутые при изучении проблемы цели, перспективы развития исследуемого вопроса

- Список использованной литературы (не меньше 10 источников), в алфавитном порядке, оформленный в соответствии с принятыми правилами. В список использованной литературы рекомендуется включать работы отечественных и зарубежных авторов, в том числе статьи, опубликованные в научных журналах в течение последних 3-х лет и ссылки на ресурсы сети Интернет.

- Приложение (при необходимости).

Требования к оформлению:

- текст с одной стороны листа;
- шрифт Times New Roman;
- кегль шрифта 14;
- межстрочное расстояние 1,5;
- поля: сверху 2,5 см, снизу – 2,5 см, слева - 3 см, справа 1,5 см;
- доклад должен быть представлен в сброшюрованном виде.

Порядок защиты доклада:

Защита доклада проводится на практических занятиях, после окончания работы студента над ним и исправления всех недочетов, выявленных преподавателем в ходе консультаций. На защиту доклада отводится 5-7 минут времени, в ходе которого студент должен показать свободное владение материалом по заявленной теме. При защите доклада приветствуется использование мультимедиа-презентации.

Оценка доклада

Доклад оценивается по следующим критериям:

- соблюдение требований к его оформлению;
- необходимость и достаточность для раскрытия темы приведенной в тексте доклада информации;
- умение студента свободно излагать основные идеи, отраженные в докладе;
- способность студента понять суть задаваемых преподавателем и сокурсниками вопросов и сформулировать точные ответы на них.

Критерии оценки:

Оценка «отлично» выставляется студенту, если в докладе студент исчерпывающе, последовательно, четко и логически стройно излагает материал; свободно справляется с задачами, вопросами и другими видами применения знаний; использует для написания доклада современные научные материалы; анализирует полученную информацию; проявляет самостоятельность при написании доклада.

Оценка «хорошо» выставляется студенту, если качество выполнения доклада достаточно высокое. Студент твердо знает материал, грамотно и по существу излагает его, не допуская существенных неточностей в ответе на вопросы по теме доклада.

Оценка «удовлетворительно» выставляется студенту, если материал доклада излагается частично, но пробелы не носят существенного характера, студент допускает неточности и ошибки при защите доклада, дает недостаточно правильные формулировки, наблюдаются нарушения логической последовательности в изложении материала.

Оценка «неудовлетворительно» выставляется студенту, если он не подготовил доклад или допустил существенные ошибки. Студент неуверенно излагает материал доклада, не отвечает на вопросы преподавателя.

4.5. Методические рекомендации по выполнению исследовательских проектов

Исследовательская проектная работа – это групповая работа, для выполнения которой необходим выбор и приложение научной методики к поставленной задаче, получение собственного теоретического или экспериментального материала, на основании которого необходимо провести анализ и сделать выводы об исследуемом явлении. Выполнение проекта – это всегда коллективная, творческая практическая работа, предназначенная для получения определенного продукта или научно-технического результата. Такая работа подразумевает четкое, однозначное формирование поставленной задачи, определение сроков выполнения намеченного, определение требований к разрабатываемому объекту.

Выполнение 1 группового проекта является обязательным условием выполнения самостоятельной работы по любой дисциплине профессионального цикла. Тема проектного задания может быть выбрана студентом из предложенных в рабочей программе или фонде оценочных средств дисциплины, либо определена самостоятельно, исходя из интересов студента (в рамках изучаемой дисциплины). Выбранную тему необходимо согласовать с преподавателем.

Требования по выполнению и оформлению проекта

При выполнении проекта приветствуется работа в группе (2-3 человека). Проект – это исследовательская работа, в ходе которой студенты должны продемонстрировать владение навыками научного исследования, умения проводить анализ, обобщать информацию, делать выводы, предлагать свои решения проблемы, рассматриваемой в проекте.

При подготовке материалов проекта студенты должны продемонстрировать владение современными методами компьютерной обработки данных.

Критерии оценки работы участника проекта.

Для каждого из участников проекта оцениваются:

- профессиональные теоретические знания в соответствующей области;
- умение работать со справочной и научной литературой, осуществлять поиск необходимой информации в Интернет;
- умение работать с техническими средствами;
- умение пользоваться соответствующими выполняемому проекту информационными технологиями;
- умение готовить материалы проекта для презентации: составлять и редактировать тексты, формировать презентацию проекта;
- умение работать в команде;
- умение публично представлять результаты собственной деятельности;
- коммуникабельность, инициативность, творческие способности.

Критерии выставления оценки участникам проекта

Оценка	Профессиональные компетенции	Компетенции, связанные с использованием соответствующих выполняемому проекту технических средств и информационных технологий	Иные универсальные компетенции (коммуникабельность, инициативность, умение работать в «команде», управленческие навыки и т.д.)	Отчетность
«Отлично»	Работа выполнена на высоком профессиональном уровне. Представленный	Технические средства и информационные технологии освоены и	Студент проявил инициативу, творческий подход, способность к	Проект представлен полностью и в срок.

Оценка	Профессиональные компетенции	Компетенции, связанные с использованием соответствующих выполняемому проекту технических средств и информационных технологий	Иные универсальные компетенции (коммуникабельность, инициативность, умение работать в «команде», управленческие навыки и т.д.)	Отчетность
	материал в основном фактически верен, допускаются негрубые фактические неточности. Студент свободно отвечает на вопросы, связанные с проектом.	использованы для реализации проекта полностью	выполнению сложных заданий, навыки работы в коллективе, организационные способности.	
«Хорошо»	Работа выполнена на достаточно высоком профессиональном уровне. Допущено до 4–5 фактических ошибок. Студент отвечает на вопросы, связанные с проектом, но недостаточно полно.	Обнаруживаются некоторые ошибки в использовании соответствующих технических средств и информационных технологий	Студент достаточно полно, но без инициативы и творческих находок выполнил возложенные на него задачи.	Проект представлен достаточно полно и в срок, но с некоторыми недоработками.
«Удовлетворительно»	Уровень недостаточно высок. Допущено до 8 фактических ошибок. Студент может ответить лишь на некоторые из заданных вопросов, связанных с проектом.	Обнаруживает недостаточное владение навыками работы с техническими средствами и соответствующим и информационным и технологиями	Студент выполнил большую часть возложенной на него работы.	Проект сдан со значительным опозданием (более недели) и не полностью
«Неудовлетворительно»	Работа не выполнена или выполнена на низком уровне. Допущено более 8 фактических ошибок. Ответы на связанные с проектом вопросы обнаруживают непонимание предмета и отсутствие	Навыков работы с техническими средствами нет, информационные технологии не освоены	Студент практически не работал, не выполнил свои задачи или выполнил лишь отдельные не существенные поручения в групповом проекте.	Проект не сдан.

Оценка	Профессиональные компетенции	Компетенции, связанные с использованием соответствующих выполняемому проекту технических средств и информационных технологий	Иные универсальные компетенции (коммуникабельность, инициативность, умение работать в «команде», управленческие навыки и т.д.)	Отчетность
	ориентации в материале проекта.			

Студенты должны: защитить проект в режиме презентации, предъявить файлы выполненного проекта, уметь рассказать о технологиях, использованных ими при выполнении проекта, дать оценку работы каждого члена группы (если проект групповой).

4.6. Методические рекомендации по подготовке к экзаменам и зачетам

Изучение многих общепрофессиональных и специальных дисциплин завершается экзаменом. Подготовка к экзамену способствует закреплению, углублению и обобщению знаний, получаемых, в процессе обучения, а также применению их к решению практических задач. Готовясь к экзамену, студент ликвидирует имеющиеся пробелы в знаниях, углубляет, систематизирует и упорядочивает свои знания. На экзамене студент демонстрирует то, что он приобрел в процессе обучения по конкретной учебной дисциплине.

Экзаменационная сессия - это серия экзаменов, установленных учебным планом. Между экзаменами интервал 3-4 дня. Не следует думать, что 3-4 дня достаточно для успешной подготовки к экзаменам.

В эти 3-4 дня нужно систематизировать уже имеющиеся знания. На консультации перед экзаменом студентов познакомят с основными требованиями, ответят на возникшие у них вопросы. Поэтому посещение консультаций обязательно.

Требования к организации подготовки к экзаменам те же, что и при занятиях в течение семестра, но соблюдаться они должны более строго. Во-первых, очень важно соблюдение режима дня; сон не менее 8 часов в сутки, занятия заканчиваются не позднее, чем за 2-3 часа до сна. Оптимальное время занятий - утренние и дневные часы. В перерывах между занятиями рекомендуются прогулки на свежем воздухе, неустойчивые занятия спортом. Во-вторых, наличие хороших собственных конспектов лекций. Даже в том случае, если была пропущена какая-либо лекция, необходимо во время ее восстановить (переписать ее на кафедре), обдумать, снять возникшие вопросы для того, чтобы запоминание материала было осознанным. В-третьих, при подготовке к экзаменам у студента должен быть хороший учебник или конспект литературы, прочитанной по указанию преподавателя в течение семестра. Здесь можно эффективно использовать листы опорных сигналов.

Вначале следует просмотреть весь материал по сдаваемой дисциплине, отметить для себя трудные вопросы. Обязательно в них разобраться. В заключение еще раз целесообразно повторить основные положения, используя при этом листы опорных сигналов.

Систематическая подготовка к занятиям в течение семестра позволит использовать время экзаменационной сессии для систематизации знаний.

Контроль самостоятельной работы студентов

Контроль самостоятельной работы проводится преподавателем в аудитории.

Предусмотрены следующие виды контроля: собеседование, оценка доклада, оценка презентации, оценка участия в круглом столе, оценка выполнения проекта.

Подробные критерии оценивания компетенций приведены в Фонде оценочных средств для проведения текущей и промежуточной аттестации.

Список литературы для выполнения СРС

Перечень основной литературы:

1. Джошуа, Блох. Java. Эффективное программирование / Блох Джошуа : практическое пособие ; пер. В. Стрельцов ; ред. Р. Усманов Электронный ресурс. - Java. Эффективное программирование, 2019-04-19 : Профобразование ; Саратов, 2019. - 310 с. - Книга находится в базовой версии ЭБС IPRbooks. - ISBN 978-5-4488-0127-3, экземпляров неограничено.

2. Мейер, Б. Объектно-ориентированное программирование и Информационные системы и технологии : учебное пособие / Мейер Б. - Москва : Интернет-Университет Информационных Технологий (ИНТУИТ), 2019. - 285 с. - Книга находится в базовой версии ЭБС IPRbooks., экземпляров неограничено.

Перечень дополнительной литературы:

1. Васильев, А. Н. Java. Объектно-ориентированное программирование : [учеб. пособие] / А.Н. Васильев. - СПб. : Питер, 2019. - 400 с. : ил. - (Учебное пособие). - Прил.: с. 379-395. - Библиогр.: с. 377. - ISBN 978-5-459-01050-3, экземпляров.

2. Чеповский, А. Common Intermediate Language и системное программирование в Microsoft .NET / А. Чеповский ; А. Макаров ; С. Скоробогатов. - 2-е изд., исправ. - Москва : Национальный Открытый Университет «ИНТУИТ», 2020. - 399 с.

Методическая литература:

1. Методические рекомендации для самостоятельной работы студентов по дисциплине «Объектно-ориентированное программирование»
2. Методические указания к лабораторным работам по дисциплине «Объектно-ориентированное программирование»

Интернет-ресурсы:

1. <http://el.ncfu.ru/> – система управления обучением ФГАОУ ВО СКФУ. Дистанционная поддержка дисциплины «Объектно-ориентированное программирование».

2. <http://metanit.com> - сайт посвящен различным языкам и технологиям программирования, компьютерам, мобильным платформам и ИТ-технологиям