



Министерство науки и высшего образования Российской Федерации  
Рубцовский индустриальный институт (филиал)  
ФГБОУ ВО «Алтайский государственный технический  
университет им. И.И. Ползунова»  
Кафедра прикладной математики

**Л.А. ПОПОВА**

**ПРОГРАММИРОВАНИЕ ПРИЛОЖЕНИЙ  
(ЧАСТЬ 1)**

Учебно-методические указания для студентов направления  
«Информатика и вычислительная техника»

Рубцовск 2021

ББК 32.973.2

Попова Л.А. Программирование приложений (часть 1): Учебно-методические указания для студентов направления «Информатика и вычислительная техника» / Л.А. Попова. – Рубцовск: РИИ, 2021. – 60 с. [ЭР].

Учебно-методические указания предназначены для проведения практических и лабораторных работ по курсу «Программирование приложений» у студентов направления 09.03.01 «Информатика и вычислительная техника» очной и заочной форм обучения.

Приведены задания к практическим и лабораторным работам, а также вопросы для самостоятельной подготовки к текущему и промежуточному тестированию.

Рассмотрены и одобрены на заседании кафедры прикладной математики Рубцовского индустриального института.  
Протокол № 9 от 18.03.2021 г.

## Содержание

Введение .....	4
1 Основы разработки приложений на языке C# в среде Visual Studio.....	4
Лабораторная работа 1 .....	8
Лабораторная работа 2 .....	8
2 Условные операторы .....	10
Лабораторная работа 3 .....	12
3 Операторы (инструкции) циклов .....	13
Лабораторная работа 4 .....	17
Лабораторная работа 5 .....	20
4 Одномерные массивы .....	22
Лабораторная работа 6 .....	26
Лабораторная работа 7 .....	28
5 Многомерные массивы .....	29
Лабораторная работа 8 .....	30
6 Коллекции.....	32
Лабораторная работа 9 .....	35
7 Организация системы ввода-вывода .....	37
Лабораторная работа 10 .....	42
Лабораторная работа 11 .....	43
8 Классы и объекты .....	43
Лабораторные работы 12-13.....	45
Лабораторная работа 14 .....	45
9 Задания для практических занятий и самостоятельной работы .....	47
10 Темы для расчетного задания (для студентов очной формы обучения) или контрольной работы (для студентов заочной формы обучения) .....	52
Список использованной литературы.....	60

## Введение

Учебно-методические указания написаны в соответствии с программой дисциплины «Программирование приложений» для студентов направления «Информатика и вычислительная техника» очной и заочной форм обучения, предназначены для аудиторной и самостоятельной работы по данному курсу.

Указания содержат краткий теоретический материал по основам программирования на языке С#, примеры решения задач, задания к лабораторным работам и практическим занятиям, а также вопросы для самостоятельной работы и подготовки к текущему контролю успеваемости и промежуточной аттестации.

### Перечень планируемых результатов обучения по дисциплине, соотнесенных с индикаторами достижения компетенций

Компетенция	Содержание компетенции	Индикатор	Содержание индикатора
ПК-5	Способен разрабатывать требования и проектировать программное обеспечение	ПК-5.1	Применяет выбранные языки программирования для написания программ
ПК-3	Способен проектировать пользовательские интерфейсы по готовому образцу или концепции интерфейса	ПК-3.1	Проектирует интерфейс по концепции или по образцу уже спроектированной части интерфейса

Общий объем дисциплины во 2 семестре: 5 з.е. / 180 час

Форма промежуточной аттестации: Экзамен

## 1 Основы разработки приложений на языке С# в среде Visual Studio

### 1.1 Среда NET Framework

.NET Framework – программная платформа, выпущенная компанией Microsoft в 2002 году. В настоящее время .NET Framework получает развитие в виде .NET Core, изначально предполагающей кроссплатформенную разработку и эксплуатацию.

*Назначение* .NET Framework – служить средой для поддержки разработки и выполнения сильно распределенных компонентных приложений. Она обеспечивает совместное использование разных языков программирования, а также безопасность, переносимость программ и общую модель программирования для платформы Windows.

NET Framework также выполняет роль среды исполнения. *Среда исполнения* – это словно некая виртуальная машина или песочница, в котором приложение работает. В .NET эта среда называется *Common Language Runtime*.

Когда пользователь запускает приложение, его код компилируется в машинный код внутри среды исполнения, после чего собственно и исполняется. CLR также предоставляет разработчикам другие сервисы, вроде управления памятью, потоками процессора, программными исключениями и безопасностью. Среда исполнения – это «прослойка» между приложением и железом, на котором оно работает.

Как правило, при написании программы на C# формируется, так называемый, *управляемый* код. Как пояснялось выше, такой код выполняется под управлением среды CLR, и поэтому на него накладываются определенные ограничения, хотя это и дает ряд преимуществ.

*Портативность* – один из самых больших плюсов использования среды исполнения. Разработчик может написать код с использованием любого из поддерживаемых языков, вроде C#, C++, Visual Basic и так далее. Этот код будет работать на любом железе, которое поддерживает .NET. Хотя платформа была создана с целью работать на разном железе (не только на Windows-компьютерах), проприетарная натура .NET Framework привела к тому, что его используют только в Windows-приложениях.

*Библиотека классов* является комплексной объектно-ориентированной коллекцией повторно используемых типов, которые применяются для разработки приложений – начиная с обычных приложений, запускаемых из командной строки, и приложений с графическим интерфейсом (GUI) и заканчивая приложениями, использующими последние технологические возможности ASP.NET, такие как веб-формы и веб-службы XML.

Development Frameworks (библиотеки разработки) включает такие библиотеки, как WPF (Windows Presentation Foundation), ASP.NET, Entity Framework, WCF (Windows Communication Foundation), Windows Store и др. Фактически это тоже классы. Отличие заключается в том, что эти классы предназначены для решения специфических задач.

## 1.2 Создание приложения в среде программирования Visual Studio

Скачать Visual Studio Community можно с официального сайта [visualstudio.com](http://visualstudio.com). Она бесплатна для обучения, для индивидуальных разработчиков, для работы над Open Source проектами и для научных исследований.

В состав Visual Studio входят:

1. текстовый редактор с синтаксической подсветкой кода;
2. система помощи IntelliSense, которая вызывается автоматически или сочетанием клавиш Ctrl + Space (пробел);
3. компиляторы с разных языков;
4. средства быстрой разработки (RAD – Rapid Application Development);
5. визуальный дизайнер интерфейсов, диаграмм;
6. компонент работы с серверами, с базами данных;
7. web-сервер IIS и sql-сервер Express варианта;

8. отладчики, профилировщики, компоненты позволяющие обрабатывать ошибки;
9. система помощи MSDN.

Для создания проекта в Visual Studio существует понятие шаблона проекта.

**Проект** – это единица компиляции. Он состоит из набора файлов. Проект компилируется целиком обычно в сборку (программу exe-файл, либо библиотеку dll-файл).

**Решение** (solution) – это несколько проектов, объединенные общими библиотеками и задачами. Как правило, открывать с помощью Visual Studio нужно именно файл решения (.sln), хотя можно открыть и отдельный проект (.csproj файл). Имейте в виду, если открыть отдельный кодовый файл, не открывая проект или решение, то не будет возможности его запустить. Это распространённая ошибка новичков.

**Ссылки** (reference) содержит те библиотеки, которые использует приложение. Только сославшись на другую сборку, можно будет использовать код из неё. Они по умолчанию включаются в каждый проект. Их можно удалить.

**Кодовый файл** – это один из файлов (с расширением .cs) на языке C#.

**Пространство имен** – это совокупность классов, логически связанных между собой.

**Класс** – это совокупность данных и методов. Все сборки состоят из скомпилированных классов.

**Метод** – это последовательность действий. Аналог функций, процедур и подпрограмм в других языках. В устной речи часто используют все эти слова как синонимы, но в спецификации на язык C# используется термин «метод».

Между сборками и пространствами имен нет прямого соответствия: в сборке может храниться несколько пространств имен, а разные классы одного пространства имен могут быть определены в разных сборках.

После успешной компиляции, в директории проекта создается поддиректория bin/Debug, в которой и оказывается сборка – результат компиляции – exe или dll файлы программы.

**Панка Properties** содержит кодовый файл *AssemblyInfo.cs*, который используется для настройки приложения.

**App.config** тоже содержит настройки приложения.

**System** – библиотека, которая поставляется вместе с Net Framework.

Базовым строительным блоком программы являются **инструкции** (statement). Инструкция представляет некоторое действие, например, арифметическую операцию, вызов метода, объявление переменной и присвоение ей значения. В конце каждой инструкции в C# ставится точка с запятой (;). Данный знак указывает компилятору на конец инструкции. Например:

```
Console.WriteLine("Привет");
```

Набор инструкций может объединяться в блок кода. **Блок кода** заключается в фигурные скобки, а инструкции помещаются между открывающей и закрывающей фигурными скобками. Одни блоки кода могут содержать другие блоки.

**Main** – это метод, который определяет последовательность действий.

Является *точкой входа* в программу на языке C#. По умолчанию метод Main размещается в классе Program.

Название класса может быть любым. Но метод Main является обязательной частью консольного приложения. Если мы изменим его название, то программа не скомпилируется.

Минимальная программа должна содержать хотя бы один класс. Пространство имен в этом случае не обязательно.

*Статический класс* может использоваться как обычный контейнер для наборов методов, работающих на входных параметрах, и не должен возвращать или устанавливать каких-либо внутренних полей экземпляра. Например, в библиотеке классов .NET Framework статический класс System.Math содержит методы, выполняющие математические операции, без требования сохранять или извлекать данные, уникальные для конкретного экземпляра класса Math. Это значит, что члены класса применяются путем задания имени класса и имени метода.

Выражение using static подключает в программу все статические методы и свойства, а также константы. И после этого мы можем не указывать название класса при вызове метода.

Подобным образом можно определять свои классы и импортировать их.

Для различных классов мы можем использовать *псевдонимы*. Затем в программе вместо названия класса используется его псевдоним.

Наиболее часто используемыми являются методы чтения и записи информации.

Для вывода информации в окно консоли используются два основных метода:

Write() – для вывода информации без перевода строки;

WriteLine() – для вывода информации с переводом строки.

В качестве аргумента этим функциям может передаваться текстовая строка для вывода, а также объект других стандартных типов данных, который по умолчанию будет преобразован к строковому представлению.

### 1.3 Операторы и выражения

Код программы содержит операции и вызовы методов. Каждая **операция** включает в себя операнды и операторы, которые определяют требуемые действия. Каждая операция или вызов функции заканчивается символом ";". Если одна строка содержит несколько операторов, то такая строка называется **выражением**. Выражение может содержать несколько операндов и операторов, и их порядок выполнения внутри выражения задается **приоритетом**. Если выражение содержит операторы одинакового приоритета, то их выполнение осуществляется слева направо в порядке следования.

## Лабораторная работа 1

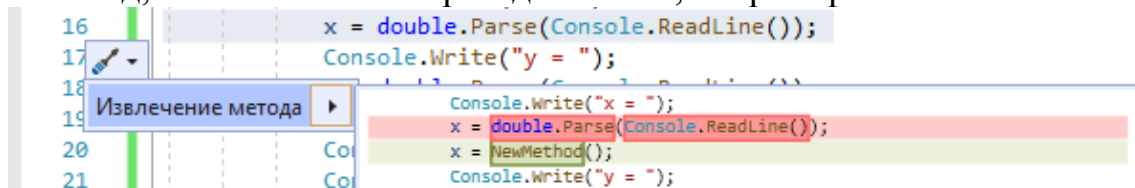
### Типы данных языка программирования C#

- 1 Запустите среду программирования Visual Studio.
- 2 Создайте проект, тип – *Консольное приложение C#* (.NET Framework).
- 3 Сохраните проект в своей папке (имя произвольное).
- 4 Объясните каждую строчку программы-заготовки.
- 5 Добавьте вывод произвольного текста.
- 6 Запустите программу на выполнение (F5 или Ctrl + F5).
- 7 Составьте программу для определения и вывода объема памяти (sizeof), диапазонов допустимых значений (MinValue и MaxValue) каждого стандартного типа C# и типа перечисления.  
*Примечание:* должны быть использованы короткие имена и CLR-типы.
- 8 Опишите переменные различных типов, инициализируйте их значениями (соответствующих типов) и выведите эти значения.
- 9 Опишите несколько переменные с использованием служебного слова var, инициализировав их значениями выражений. Определите типы этих переменных.
- 10 Подготовьте ответы по следующим темам:
  - Структура проекта программы на C#.
  - Назначение директивы using. Когда следует использовать using static?
  - Типы данных.

## Лабораторная работа 2

### Работа в среде Visual Studio

- 1 Составьте программу для нахождения значения функции:  $z = \frac{(x-1)^3 + 5e^{-y}}{\sqrt{|bx| + y^4 + 1}}$ , где  $b = 3$  – целочисленная константа;  $x, y, z$  – переменные. Для этого:
  - 1.1 Составьте контрольный пример.
  - 1.2 Создайте консольное приложение.
  - 1.3 Введите текст программы.
- 2 Запустите программу на выполнение. Проверьте правильность работы на контрольном примере.
- 3 Измените текст программы, выделив из него три метода: *ввод* данных, *вычисление* значения функции, *вывод* результата. Для этого:
  - 3.1 Сделайте константу  $b$  глобальной для класса (видимой во всех методах).
  - 3.2 Выделите строку с программным кодом, из которой надо извлечь метод, и вызовите «Быстрые действия», например:



```
16 x = double.Parse(Console.ReadLine());
17 Console.Write("y = ");
18
19 Console.Write("x = ");
20 x = double.Parse(Console.ReadLine());
21 x = NewMethod();
22 Console.Write("y = ");
```



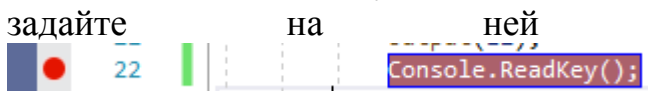
3.3 Измените стандартные названия методов в соответствии с их назначением.

3.4 Переименуйте переменные в методе **Main** (имена фактических и формальных параметров должны отличаться).

4 Запустите программу в режиме отладчика. Для этого:

10.1 Добавьте команду *Console.ReadKey()* в конце метода **Main** и

задайте 22 на ней точку останова:



4.1 Запустите программу в режиме отладки (кнопка «Пуск», F5). Введите произвольные числовые данные.

*Примечание:* Выполнение программы приостановится, когда будет достигнута точка останова (перед выполнением метода `Console.ReadKey`). В окне **Видимые** отображаются значения всех переменных, которые используются рядом с текущей строкой. В окне **Локальные** отображаются значения переменных, которые определены в текущем выполняемом методе.

4.2 В *окне интерпретации* выполните вычисление значения функции  $z(x, y)$  с разными аргументами (введите имя функции и в скобках значения параметров).

5 Добавить строки документации для методов.

6 Составьте контрольный пример и программу в соответствии с вашим вариантом. Запустите на выполнение и отладьте ее.

7 Подготовьте ответы по следующим темам:

- Назначение и возможности режима отладки программы на C#.
- Методы: назначение и структура.
- Глобальные и локальные переменные. Область видимости переменных.

Задания по вариантам:

Вычислить значение функции ( $a, b$  – константы):

$$1) \quad y = x^5 - \frac{x^3}{\sin^2 x} + 2\sqrt{ax^2 + 1}$$

$$2) \quad y = 3x^2 \cdot \sqrt[3]{x+1} + ae^{3x}$$

$$3) \quad y = ax + \frac{b}{e^{x+1}} + \sqrt[3]{x^2 + \sin x}$$

$$4) \quad y = 2 \cdot \sqrt{|x| + b} - \ln(x + e^{2x})$$

$$5) \quad y = a^x \cos(2x) - \frac{1}{|x| + 1}$$

$$6) \quad y = \ln(2x^3 + b) + \sqrt{|a - 2x^3|}$$

$$7) \quad y = \frac{\cos^2 x - \sin^2 ax}{1 + 3 \ln x}$$

$$8) \quad y = \frac{\sqrt{|ax+1|}}{2} - \frac{e^x}{\sqrt[3]{ax+1}}$$

$$9) \quad y = \frac{x \cos ax}{\sqrt[3]{x^3+3x+1}}$$

$$10) \quad y = \sin\left(\frac{3a}{2} + x\right) + b\sqrt{|x|} + e^x$$

## 2 Условные операторы

### 2.1 Оператор условие

**Оператор условие** – единственная операция языка Си, имеющая *три операнда*. Эта операция имеет вид:

выражение1 ? выражение2 : выражение3

Вычисляется выражение1. Если оно имеет значение true, то результатом операции будет значение выражения2. Иначе значение выражения3.

Например, операцию условие можно применять для нахождения наибольшего из двух чисел x и y:

max = (x > y) ? x : y;

или для нахождения абсолютной величины числа x:

abs = (x > 0) ? x : -x;

### 2.2 Оператор (инструкция) if

Для организации условного ветвления язык C# унаследовал от C и C++ конструкцию if...else (если ... иначе).

Полный формат ее записи такой:

**if** (условие) инструкция1; **else** инструкция2;

Здесь под элементом инструкция понимается одна инструкция языка C#. Часть *else* необязательна. Если по каждому из условий нужно выполнить более одного оператора, эти операторы должны быть объединены в блок с помощью фигурных скобок {...}.

В отличие от языков C и C++, в C# условный оператор *if* может работать только с булевыми выражениями, но не с произвольными значениями вроде -1 и 0.

Если элемент условие, который представляет собой условное выражение, при вычислении даст значение ИСТИНА, будет выполнена if-инструкция1; в противном случае – else-инструкция2 (если такая существует).

Вложенные условные операторы можно представить в следующем виде:

```
if (условие) инструкция1;
else if (условие) инструкция2;
else if (условие) инструкция3;
else инструкция4;
```

### 2.3 Оператор (инструкция) switch

Является оператором выбора. Инструкция switch обеспечивает многонаправленное ветвление. Она позволяет делать выбор одной из множества альтернатив. Хотя многонаправленное тестирование можно

реализовать с помощью последовательности вложенных if-инструкций, для многих ситуаций инструкция switch оказывается более эффективным решением. Общий формат записи инструкции switch:

```
switch (выражение)
{
    case константа 1:
        последовательность инструкций; break;
    case константа2:
        последовательность инструкций; break;
    case константа3:
        последовательность инструкций; break;
    default:
        последовательность инструкций; break;
}
```

Элемент выражение инструкции switch должен иметь целочисленный тип (например, char, byte, short или int) или тип string. Выражения, имеющие тип с плавающей точкой, не разрешены. Очень часто в качестве управляющего switch-выражения используется просто переменная; case-константы должны быть литералами, тип которых совместим с типом заданного выражения. При этом никакие две case-константы в одной switch-инструкции не могут иметь идентичных значений.

Оператор работает следующим образом. Значение выражения последовательно сравнивается с константами из заданного списка. При обнаружении совпадения для одного из условий сравнения выполняется последовательность инструкций, связанная с этим условием. Последовательность инструкций default-ветви выполняется в том случае, если ни одна из заданных case-констант не совпадет с результатом вычисления switch-выражения. Ветвь default необязательна. Если она отсутствует, то при несовпадении результата выражения ни с одной из case-констант, никакое действие выполнено не будет. Если такое совпадение все-таки обнаружится, будут выполнены инструкции, соответствующие данной case-ветви до тех пор, пока не встретится инструкция break.

"Пустые" case-инструкции могут "проваливаться".

```
using System;
class EmptyCasesCanFall
{
    public static void Main()
    {
        int i;
        for (i = 1; i < 5; i++) switch (i)
        {
            case 1:
            case 2:
            case 3: Console.WriteLine("i равно 1, 2 или 3"); break;
            case 4: Console.WriteLine("i равно 4"); break;
        }
    }
}
```

## Лабораторная работа 3

### Реализация разветвляющихся алгоритмов на языке C#

- 1 Решите задачи в соответствии с вашим вариантом (первое задание с использованием условного оператора, второе задание с использованием оператора выбора).
- 2 Подготовьте ответы по следующим темам:
  - Правила записи логических выражений и вычисления их значений.
  - Синтаксис и семантика условного оператора.
  - Синтаксис и семантика оператора выбора.

#### *Вариант 1*

1) Требуется определить, бьет ли слон, стоящий на клетке с указанными координатами (номер строки и номер столбца), фигуру, стоящую на другой указанной клетке. Вводятся четыре числа: координаты слона и координаты другой фигуры (данные введены корректно).

2) Для всех цифр вывести их словесное название.

#### *Вариант 2*

1) Определить, в какой координатной четверти расположен треугольник, образованный прямой, заданной уравнением  $y = ax + b$ , и осями координат.

2) По введенному названию месяца определить и вывести количество дней в нем.

#### *Вариант 3*

1) Дано четырехзначное число. Определить, является ли оно симметричным.

2) По введенному номеру месяца определить и вывести название времени года.

#### *Вариант 4*

1) Требуется определить, бьет ли король, стоящий на клетке с указанными координатами (номер строки и номер столбца), фигуру, стоящую на другой указанной клетке. Вводятся четыре числа: координаты короля и координаты другой фигуры (данные введены корректно).

2) По первой букве из набора КОЖЗГСФ определить и вывести название цвета радуги.

#### *Вариант 5*

1) Определить, можно ли от шоколадки размером  $n \times m$  долек отломить  $k$  долек, если разрешается сделать один разлом по прямой между дольками (то есть разломить шоколадку на два прямоугольника, площадь одного из них равна  $k$ ).

2) Дана градусная мера угла на плоскости (от 85 до 95). Определить к какому типу относится данный угол (острый, тупой или прямой).

### Вариант 6

1) Даны три числа  $a, b, c$ . Упорядочить их значения в порядке неубывания (должно выполняться условие  $a \leq b \leq c$ ).

2) По введенному номеру недели и названию дня недели вывести количество пар в этот день или сообщение "выходной день".

### Вариант 7

1) Решить уравнение  $(ax + b) / (cx + d) = 0$ .

2) Дано натуральное число, состоящее только из единиц. Определить, является ли введенное число однозначным, двузначным, трехзначным, четырехзначным или содержит большее количество цифр.

### Вариант 8

1) Требуется определить, бьет ли конь, стоящий на клетке с указанными координатами (номер строки и номер столбца), фигуру, стоящую на другой указанной клетке. Вводятся четыре числа: координаты коня и координаты другой фигуры (данные введены корректно).

2) По введенному с клавиатуры символу из алфавита  $A = \{ 't', 'q', 's', 'f', '1', '3', '5', 'ф', 'б', 'ю', ':', '%' \}$  определить, к какой категории он относится (буква русского алфавита, буква английского алфавита, цифра или специальный символ).

### Вариант 9

1) Среди трех введенных чисел найти среднее значение.

2) Дано целое число  $k$  (0-99), определяющее цену товара. Вывести сообщение, добавляющее к значению  $k$  слово "рубль", "рубля" или "рублей". Например, при  $k = 32$  вывести "Цена товара составляет 32 рубля".

### Вариант 10

1) Дан кирпич (прямоугольный параллелепипед) со сторонами  $a, b, c$  и прямоугольное отверстие  $n \times m$ . Требуется определить, может ли кирпич пройти через отверстие.

2) По введенной температуре воды в градусах Цельсия (от  $-30^\circ$  до  $130^\circ$  с шагом  $20^\circ$ ) определить ее агрегатное состояние (твердое, жидкое, газообразное) или вывести сообщение, что такое значение недопустимо.

### Вариант 11

1) Дано целое число. Найти число, большее заданного и ближайшее к нему, которое кратно 3.

2) Даны два целых числа и символ из множества  $(+, -, *, /, \%)$ . Вывести результат выполнения соответствующей операции над данными числами.

## 3 Операторы (инструкции) циклов

### 3.1 Цикл *for*

Этот цикл является мощным и гибким средством программирования. Начнем с традиционных форм его использования. Итак, общий формат записи цикла *for* для повторного выполнения одной инструкции имеет следующий вид:

*for* (инициализация; условие; итерация) инструкция;

**инициализация** – это выражение, вычисляемое перед первым выполнением тела цикла (обычно инициализация локальной переменной в качестве счетчика цикла). Инициализация, как правило, представлена оператором присваивания, задающим первоначальное значение переменной, которая исполняет роль счетчика и управляет циклом;

**условие** – это выражение, проверяемое перед каждой новой итерацией цикла. Элемент условие представляет собой выражение типа `bool`, в котором тестируется значение управляющей переменной цикла. Результат этого тестирования определяет, выполнится цикл `for` еще раз или нет;

**итерация** – это выражение, вычисляемое после каждой итерации (обычно приращение значения счетчика цикла).

Обратите внимание на то, что эти три основные части оператора цикла `for` должны быть разделены точкой с запятой.

Выполнение цикла `for` будет продолжаться, пока проверка условия дает истинный результат (должен возвращать `true`, чтобы была выполнена следующая итерация).

Как только эта проверка даст ложный результат, цикл завершится, а выполнение программы будет продолжено с оператора, следующего после цикла `for`.

*Объявление управляющей переменной в цикле for*

Часто переменная, которая управляет циклом `for`, необходима только для этого цикла и больше никак не используется. В этом случае можно объявить ее в разделе инициализации цикла. Например, следующая программа вычисляет как сумму, так и факториал чисел от 1 до 5. Управляющая переменная `i` здесь объявляется в цикле `for`.

```
// Объявление управляющей переменной в цикле for.
int sum = 0;
int fact = 1;
// Вычисляем сумму и факториал чисел от 1 до 5.
for (int i = 1; i <= 5; i++)
{
    sum += i; // i известна только в пределах цикла,
    fact *= i;
}
// Но здесь переменная i неизвестна.
Console.WriteLine("Сумма равна " + sum);
Console.WriteLine("Факториал равен " + fact);
```

Управляющая переменная цикла `for` может изменяться как с положительным, так и с отрицательным приращением, причем величина этого приращения также может быть любой.

*Использование нескольких управляющих переменных цикла*

Для управления циклом `for` можно использовать две или больше переменных. В этом случае инструкции инициализации и итерации для каждой из этих переменных отделяются запятыми. Вот пример:

```
for (int i = 0, j = 10; i < j; i++, --)
    Console.WriteLine("i и j: " + i + " " + j);
```

Здесь запятыми отделяются две инструкции инициализации и два итерационных выражения. При входе в цикл инициализируются обе переменные – *i* и *j*. После выполнения каждой итерации цикла переменная *i* инкрементируется, а переменная *j* декрементируется. Использование нескольких управляющих переменных в цикле иногда позволяет упростить алгоритмы. В разделах инициализации и итерации можно использовать любое количество инструкций.

Условным выражением, которое управляет циклом `for`, может быть любое допустимое выражение, генерирующее результат типа `bool`.

В `C#` разрешается опустить любой элемент заголовка цикла (инициализация, условие, итерация) или даже все сразу.

#### *Бесконечный цикл*

Оставив пустым условное выражение цикла `for`, можно создать бесконечный цикл (цикл, который никогда не заканчивается). Например, в следующем фрагменте программы показан способ, который используют многие `C#`-программисты для создания бесконечного цикла.

```
for ( ; ; ) // Специально созданный бесконечный цикл.
{
    // ...
}
```

### 3.2 Цикл `while`

Подобно `for`, *while* также является циклом с предварительной проверкой. Синтаксис его аналогичен, но циклы *while* включают только одно выражение:

***while*** (условие) инструкция;

где *оператор* – это единственный оператор или же блок операторов, а условие означает конкретное условие управления циклом и может быть любым логическим выражением.

В этом цикле оператор выполняется, пока условие истинно.

Как только условие становится ложным, управление программой передается строке кода, следующей непосредственно после цикла.

Как и в цикле *for*, в цикле *while* проверяется условное выражение, указываемое в самом начале цикла. Это означает, что код в теле цикла может вообще не выполняться, а также избавляет от необходимости выполнять отдельную проверку перед самим циклом.

```
int s = 0;
int n;
while ((n = int.Parse(Console.ReadLine())) != 0) s += n;
Console.WriteLine("s = " + s);
```

Этот тип цикла также может быть бесконечным.

### 3.3 Цикл `do...while`

Цикл *do...while* – это версия *while* с постусловием. Это значит, что условие цикла проверяется *после* выполнения тела цикла.

Следовательно, циклы *do...while* удобны в тех ситуациях, когда блок операторов должен быть выполнен как минимум *однажды*.

Ниже приведена общая форма оператора цикла *do-while*:

**do** инструкция; **while** (условие);

При наличии лишь одного оператора фигурные скобки в данной форме записи необязательны.

Тем не менее, они зачастую используются для того, чтобы сделать конструкцию *do-while* более удобочитаемой и не путать ее с конструкцией цикла *while*.

Цикл *do-while* выполняется, пока условное выражение истинно.

### 3.4 Цикл *foreach*

Цикл *foreach* служит для циклического обращения к элементам коллекции, представляющей собой группу объектов. Общая форма оператора:

**foreach** (*тип имя\_переменной\_цикла in коллекция*) инструкция;

Здесь *тип имя\_переменной\_цикла* обозначает тип и имя переменной управления циклом, которая получает значение следующего элемента коллекции на каждом шаге выполнения цикла *foreach*.

Оператор цикла *foreach* работает следующим образом. Когда цикл начинается, первый элемент коллекции выбирается и присваивается переменной цикла. На каждом последующем шаге итерации выбирается следующий элемент коллекции, который сохраняется в переменной цикла.

Цикл завершается, когда все элементы окажутся выбранными.

При компиляции кода:

```
int[] myArr = new int[5];
int k = 0;
foreach (int fVar in myArr)
    fVar = k++;
```

возникает ошибка: *Невозможно присвоить «fVar» значение, поскольку он является «переменная цикла foreach».*

Это означает, в данном типе цикла изменить элемент коллекции невозможно, и следует использовать другие конструкции цикла.

### 3.5 Управляющие операторы

С помощью оператора *break* можно специально организовать немедленный выход из цикла в обход любого кода, оставшегося в теле цикла, а также минуя проверку условия цикла.

Оператор *break* можно применять в любом цикле, предусмотренном в C#.

Если оператор *break* применяется в целом ряде вложенных циклов или внутри оператора *switch*, то он прерывает выполнение только самого внутреннего цикла.

С помощью оператора *continue* можно организовать преждевременное завершение шага итерации цикла в обход обычной структуры управления циклом.

Оператор *continue* осуществляет принудительный переход к следующему шагу цикла, пропуская любой код, оставшийся невыполненным. В циклах *while* и *do-while* оператор *continue* вызывает передачу управления непосредственно условному выражению, после чего продолжается процесс выполнения цикла. А



в цикле *for* сначала вычисляется итерационное выражение, затем условное выражение, после чего цикл продолжается.

## Лабораторная работа 4

### Реализация циклических алгоритмов с использованием инструкции *for* на языке C#

- 1) Решите задачи в соответствии с вашим вариантом.
- 2) Подготовьте ответы по следующим темам:
  - Назначение и возможности циклических алгоритмов.
  - Синтаксис и семантика цикла *for*.

#### Вариант 1

1) Дана последовательность целых чисел. Найти среднее арифметическое всех четных отрицательных чисел из последовательности.

2) Составить таблицу значений кусочно-заданной функции  $y = f(x)$  на отрезке  $[k; n]$  с шагом изменения аргумента  $dx=0.3$ :

$$y = \begin{cases} 2ax + |a - x^4|, & x < 0, \\ \frac{e^x}{\sqrt{10 + ax^2}} - 1, & 0 \leq x < 3, \\ 4x^3 - 2x^2 + 5, & x \geq 3, \end{cases}$$

где  $a=2.3$ .

#### Вариант 2

1) Дана последовательность целых чисел. Найти произведение всех двузначных положительных чисел, кратных 3, из последовательности и их количество.

2) Составить таблицу значений кусочно-заданной функции  $y = f(x)$  на отрезке  $[k; n]$  с шагом изменения аргумента  $dx=0.25$ :

$$y = \begin{cases} \sin^2(2x + a), & x < 1, \\ ax^2 - |3 - x^a|, & 1 \leq x \leq 2, \\ \frac{1}{e^{ax}} \cos \pi x, & x > 2, \end{cases}$$

где  $a=1.02$ .

#### Вариант 3

1) Дана последовательность целых чисел. Найти сумму кубов всех четных неотрицательных чисел из последовательности.

2) Составить таблицу значений кусочно-заданной функции  $y = f(x)$  на отрезке  $[a; b]$  с шагом изменения аргумента  $dx=0.35$ :

$$y = \begin{cases} \frac{4x - 8}{x^4 + k}, & x \leq -2, \\ \cos^2 x + kx^4, & -2 < x \leq k, \\ \sin nx \cdot \ln(2x - k^2), & x > k, \end{cases}$$

где  $n=2.8$ ;  $k=1.25$ .

#### Вариант 4

1) Дана последовательность целых чисел. Найти среднее геометрическое всех положительных чисел, кратных 5, из последовательности.

2) Составить таблицу значений кусочно-заданной функции  $y = f(x)$  на отрезке  $[a; b]$  с шагом изменения аргумента  $dx=0.5$ :

$$y = \begin{cases} \frac{2 \sin(3x - 1)}{x}, & x < -\pi/2, \\ e^{-x^2} + \cos 2kx, & -\pi/2 \leq x \leq 1, \\ \log_2(3x + k), & x > 1, \end{cases}$$

где  $k=1.5$ .

#### Вариант 5

1) Дана последовательность целых чисел. Найти сумму квадратов всех нечетных отрицательных чисел из последовательности и их количество.

2) Составить таблицу значений кусочно-заданной функции  $y = f(x)$  на отрезке  $[a; b]$  с шагом изменения аргумента  $dx=0.25$ :

$$y = \begin{cases} (1 + cx^2)^4, & x \leq 1, \\ \sin \frac{2\pi k}{x}, & 1 < x \leq 3, \\ \sqrt[3]{2x - 1}, & x > 3, \end{cases}$$

где  $c = -2$ ;  $k=1.5$ .

#### Вариант 6

1) Дана последовательность целых чисел. Найти произведение всех двузначных отрицательных чисел, кратных 7, из последовательности и их количество.

2) Составить таблицу значений кусочно-заданной функции  $y = f(x)$  на отрезке  $[k; n]$  с шагом изменения аргумента  $dx=0.3$ :

$$z = \begin{cases} \sin \lg ax - |1 - 2^x|, & x < 0, \\ x^2 + \sqrt{x - a}, & 0 \leq x \leq 3, \\ \frac{e^{ax-1}}{1 + x^4}, & x > 3, \end{cases}$$

где  $a = -3$ .

### Вариант 7

1) Дана последовательность целых чисел. Найти среднее арифметическое всех отрицательных или кратных 3 чисел из последовательности.

2) Составить таблицу значений кусочно-заданной функции  $y = f(x)$  на отрезке  $[k; n]$  с шагом изменения аргумента  $dx = 1.5$ :

$$y = \begin{cases} \pi x^3 - \frac{a}{x^4}, & x < 0, \\ \log_3(x + a\sqrt{2x+1} + c), & 0 \leq x \leq 10, \\ \frac{ax - c}{x + 2}, & x > 10, \end{cases}$$

где  $a = 7.2$ ;  $c = 1$ .

### Вариант 8

1) Дана последовательность целых чисел. Найти сумму квадратов всех четных однозначных чисел (отрицательных и положительных) из последовательности.

2) Составить таблицу значений кусочно-заданной функции  $y = f(x)$  на отрезке  $[k; n]$  с шагом изменения аргумента  $dx = 0.75$ :

$$y = \begin{cases} \cos^2 x - ax^3, & x \leq -2, \\ \sqrt{ax^2 + b \sin x + 5}, & -2 < x < 4, \\ \frac{ax + b}{2x - 7}, & x \geq 4, \end{cases}$$

где  $a = 4.7$ ;  $b = 1.5$ .

### Вариант 9

1) Дана последовательность целых чисел. Найти среднее геометрическое всех положительных чисел, кратных 5, из последовательности.

2) Составить таблицу значений кусочно-заданной функции  $y = f(x)$  на отрезке  $[k; n]$  с шагом изменения аргумента  $dx = 0.2$ :

$$y = \begin{cases} \sin^3(5x - a), & x \leq 0, \\ \frac{e^{-ax}}{ax + 1}, & 0 < x \leq 1, \\ 2 \lg(3x - 1), & x > 1, \end{cases}$$

где  $a=4.6$ .

### Вариант 10

1) Дана последовательность целых чисел. Найти сумму и произведение всех однозначных или кратных 4 чисел из последовательности.

2) Составить таблицу значений кусочно-заданной функции  $y = f(x)$  на отрезке  $[k; n]$  с шагом изменения аргумента  $dx=0.3$ :

$$y = \begin{cases} 2x^3 + |x - 1|, & x < 1, \\ 2a \cos(\pi x - 2), & 1 \leq x \leq 4, \\ a \lg x + \sqrt[3]{x^2}, & x > 4, \end{cases}$$

где  $a = -4.2$ .

### Вариант 11

1) Дана последовательность целых чисел. Найти среднее арифметическое всех нечетных положительных чисел из последовательности.

2) Составить таблицу значений кусочно-заданной функции  $y = f(x)$  на отрезке  $[k; n]$  с шагом изменения аргумента  $dx=0.25$ :

$$y = \begin{cases} 2 \cdot \sqrt[3]{x^4 - 1}, & x \leq -1, \\ \sin \sqrt{|ax|} + 2x, & -1 < x \leq 1, \\ e^{\frac{a}{x}} + \lg(x + \sin x), & x > 1, \end{cases}$$

где  $a=1.45$

## Лабораторная работа 5

### Реализация алгоритмов с использованием циклов с условием

- 1 Решите задачу в соответствии с вашим вариантом.
- 2 Подготовьте ответы по следующим темам:
  - Синтаксис и семантика цикла с *предусловием*.
  - Синтаксис и семантика цикла с *постусловием*.

### Вариант 1

Вычислить с заданной точностью  $\epsilon$  значение функции, разложенной в степенной ряд. Сравнить со значением встроеной функции.

$$y = \sin x = x - \frac{x^3}{3!} + \frac{x^5}{5!} - \frac{x^7}{7!} + \dots = \sum_{n=0}^{\infty} \frac{(-1)^n x^{2n+1}}{(2n+1)!},$$

где  $x \in (-\infty; \infty)$ .

### Вариант 2

Вычислить с заданной точностью  $\epsilon$  значение функции, разложенной в степенной ряд. Сравнить со значением встроеной функции.

$$y = \ln(1+x) = x - \frac{x^2}{2} + \frac{x^3}{3} - \frac{x^4}{4} + \dots = \sum_{n=0}^{\infty} \frac{(-1)^n x^{n+1}}{n+1},$$

где  $x \in (-1; 1]$ .

### Вариант 3

Вычислить с заданной точностью  $eps$  значение функции, разложенной в степенной ряд. Сравнить со значением встроенной функции.

$$y = \operatorname{arctg} x = x - \frac{x^3}{3} + \frac{x^5}{5} - \frac{x^7}{7} + \dots = \sum_{n=1}^{\infty} \frac{(-1)^{n-1} x^{2n-1}}{2n-1},$$

где  $x \in [-1; 1]$ .

### Вариант 4

Вычислить с заданной точностью  $eps$  значение функции, разложенной в степенной ряд. Сравнить со значением встроенной функции.

$$\begin{aligned} y = \arcsin x &= x + \frac{1}{2 \cdot 3} x^3 + \frac{1 \cdot 3}{2 \cdot 4 \cdot 5} x^5 + \frac{1 \cdot 3 \cdot 5}{2 \cdot 4 \cdot 6 \cdot 7} x^7 + \dots = \\ &= x + \sum_{n=1}^{\infty} \frac{(2n-1)!!}{(2n)!! \cdot (2n+1)} x^{2n+1}, \end{aligned}$$

где  $x \in [-1; 1]$ .

### Вариант 5

Вычислить с заданной точностью  $eps$  значение функции, разложенной в степенной ряд. Сравнить со значением встроенной функции.

$$y = e^{-x} = 1 - x + \frac{x^2}{2!} - \frac{x^3}{3!} + \dots = \sum_{n=0}^{\infty} \frac{(-1)^n x^n}{n!},$$

где  $x \in (-\infty; \infty)$ .

### Вариант 6

Вычислить с заданной точностью  $eps$  значение функции, разложенной в степенной ряд. Сравнить со значением встроенной функции.

$$\begin{aligned} y = \frac{1}{\sqrt{1+x}} &= 1 - \frac{1}{2} x + \frac{1 \cdot 3}{2 \cdot 4} x^2 - \frac{1 \cdot 3 \cdot 5}{2 \cdot 4 \cdot 6} x^3 + \dots = \\ &= 1 + \sum_{n=1}^{\infty} \frac{(-1)^n (2n-1)!!}{(2n)!!} x^n, \end{aligned}$$

где  $x \in (-1; 1]$ .

### Вариант 7

Вычислить с заданной точностью  $eps$  значение функции, разложенной в степенной ряд. Сравнить со значением встроенной функции.

$$y = a^x = 1 + x \ln a + \frac{(x \ln a)^2}{2!} + \frac{(x \ln a)^3}{3!} + \dots = \sum_{n=0}^{\infty} \frac{(x \ln a)^n}{n!},$$

где  $x \in (-\infty; \infty)$ .

#### Вариант 8

Вычислить с заданной точностью *eps* значение функции, разложенной в степенной ряд. Сравнить со значением встроенной функции.

$$y = \cosh x = 1 + \frac{x^2}{2!} + \frac{x^4}{4!} + \frac{x^6}{6!} + \dots = \sum_{n=0}^{\infty} \frac{x^{2n}}{(2n)!},$$

где  $x \in (-\infty; \infty)$ .

#### Вариант 9

Вычислить с заданной точностью *eps* значение функции, разложенной в степенной ряд. Сравнить со значением встроенной функции.

$$y = \sinh x = x + \frac{x^3}{3!} + \frac{x^5}{5!} + \frac{x^7}{7!} + \dots = \sum_{n=0}^{\infty} \frac{x^{2n+1}}{(2n+1)!},$$

где  $x \in (-\infty; \infty)$ .

#### Вариант 10

Вычислить с заданной точностью *eps* значение функции, разложенной в степенной ряд. Сравнить со значением встроенной функции.

$$y = \cos x = 1 - \frac{x^2}{2!} + \frac{x^4}{4!} - \frac{x^6}{6!} + \dots = \sum_{n=0}^{\infty} \frac{(-1)^n x^{2n}}{(2n)!},$$

где  $x \in (-\infty; \infty)$ .

#### Вариант 11

Вычислить с заданной точностью *eps* значение функции, разложенной в степенной ряд. Сравнить со значением встроенной функции.

$$y = e^x = 1 + x + \frac{x^2}{2!} + \frac{x^3}{3!} + \dots = \sum_{n=0}^{\infty} \frac{x^n}{n!},$$

где  $x \in (-\infty; \infty)$ .

## 4 Одномерные массивы

Массив (*array*) – это совокупность данных одного типа, обращение к которым происходит с использованием общего для всех имени. Массивы представляют собой удобное средство группирования связанных переменных.

C#-массивы реализованы как объекты. Это позволило неиспользуемые массивы автоматически удалять системой сбора мусора.

*Одномерный массив* – это список однотипных, связанных по смысловому содержанию данных. Для объявления одномерного массива используется следующая форма записи.

```
тип[] имя_массива = new тип[размер];
```

Здесь с помощью элемента записи *тип* объявляется базовый тип массива. *Базовый* тип определяет тип данных каждого элемента, составляющего массив. Обратите внимание на *одну* пару квадратных скобок за элементом записи *тип*. Это означает, что определяется одномерный массив. Количество элементов, которые будут храниться в массиве, определяется элементом записи *размер*. Поскольку массивы реализуются как объекты, их создание представляет собой двухступенчатый процесс. Сначала объявляется ссылочная переменная на массив, а затем для него выделяется память, и переменной массива присваивается ссылка на эту область памяти. Таким образом, в C# массивы динамически размещаются в памяти с помощью оператора `new`. Например,

```
int[] sample = new int[10];
```

Доступ к отдельному элементу массива осуществляется посредством индекса. *Индекс* описывает позицию элемента внутри массива. Поскольку массив `sample` содержит 10 элементов, его индексы изменяются от 0 до 9. Так, первым элементом массива является `sample[0]`, а последним – `sample[9]`.

#### 4.1 Выполнение действий над элементами массива

```
// Демонстрация использования одномерного массива.
using System;
class ArrayDemo
{
    public static void Main()
    {
        int[] sample = new int[10];
        for (int i = 0; i < 10; i++) sample[i] = i;
        for (int i = 0; i < 10; i++)
            Console.WriteLine("sample[" + i + "]: " + Sample[i]);
    }
}
```

Заполнение массива случайными числами:

```
private static Random rnd = new Random();
int[] A = new int[50];
for (int i = 0; i < A.Length; i++)
    A[i] = rnd.Next(1, 100);
```

Примеры ввод элементов массива:

```
int[] nums = new int[10];
for (int i = 0; i < nums.Length; i++) Console.WriteLine(nums[i]);
// или так
int[] nums = new int[10];
int j = 0;
foreach (var item in Console.ReadLine().Split())
{
    nums[j++] = int.Parse(item);
}
```

```

}
foreach (var item in nums)
    Console.WriteLine(item);

    Поиск элементов в массиве
static int FindIndex(int[] array, int number)
{
    for (int i = 0; i < array.Length; i++)
        if (array[i] == number) return i;
    return -1;
}

```

Бинарный поиск индекса первого слева элемента массива, равного заданному числу (левосторонний):

```

static int FindIndexByBinarySearch(int[] array, int number)
{
    var left = 0;
    var right = array.Length - 1;
    while (left < right)
    {
        var middle = (right + left) / 2;
        if (number <= array[middle])
            right = middle;
        else left = middle + 1;
    }
    if (array[right] == number)
        return right;
    return -1;
}

public static void Main()
{
    var array = new[] { 1, 2, 3, 4, 5, 5, 5, 6 };
    Console.WriteLine(FindIndex(array, 5));
    Console.WriteLine(FindIndexByBinarySearch(array, 5));
}

```

#### 4.2 Методы для сортировки массива

```

/*Сортировка пузырьком первый вариант*/
private static void BubbleSort(int[] array)
{
    for (int i = 0; i < array.Length; i++)
        for (int j = 0; j < array.Length - 1; j++)
            if (array[j] > array[j + 1])
            {
                int t = array[j + 1];
                array[j + 1] = array[j];
                array[j] = t;
            }
}

```



```

/*Сортировка пузырьком другой вариант*/
private static void BubbleSort1(int[] array)
{
    var flag = true;
    while (flag)
    {
        flag = false;
        for (int j = 0; j < array.Length - 1; j++)
            if (array[j] > array[j + 1])
            {
                int t = array[j + 1];
                array[j + 1] = array[j];
                array[j] = t;
                flag = true;
            }
    }
}

```

```

/*Сортировка пузырьком сокращенная*/
private static void BubbleSort2(int[] array)
{
    for (int i = array.Length - 2; i > 0; i--)
        for (int j = 0; j < i; j++)
            if (array[j] > array[j + 1])
            {
                int t = array[j + 1];
                array[j + 1] = array[j];
                array[j] = t;
            }
}

```

```

/*Линейная сортировка*/
private static void LineSort(int[] array)
{
    for (int i = 0; i < array.Length - 1; i++)
        for (int j = i + 1; j < array.Length; j++)
            if (array[i] > array[j])
            {
                int t = array[j];
                array[j] = array[i];
                array[i] = t;
            }
}

```

```

/*Сортировка выбором*/
private static void CaseSort(int[] array)
{
    for (int i = 0; i < array.Length - 1; i++)
    {
        int min = i;

```

```

        for (int j = i + 1; j < array.Length; j++)
            if (array[min] > array[j]) min = j;
        int t = array[min];
        array[min] = array[i];
        array[i] = t;
    }
}

/*Быстрая сортировка (QuickSort) и сортировка Хоара (HoareSort) – это синонимы*/
static void HoareSort(int[] array, int start, int end)
{
    if (end == start) return;
    var pivot = array[end]; //опорный элемент
    var storeIndex = start;
    for (int i = start; i < end; i++)
        if (array[i] <= pivot)
        {
            var t = array[i];
            array[i] = array[storeIndex];
            array[storeIndex] = t;
            storeIndex++;
        }

    var n = array[storeIndex];
    array[storeIndex] = array[end];
    array[end] = n;
    if (storeIndex > start) HoareSort(array, start, storeIndex - 1);
    if (storeIndex < end) HoareSort(array, storeIndex + 1, end);
}

static void HoareSort(int[] array)
{
    HoareSort(array, 0, array.Length - 1);
}

using System.Diagnostics;
var watch = new Stopwatch();
watch.Start();
// здесь надо записать тестируемый код
watch.Stop();
Console.WriteLine(watch.ElapsedMilliseconds);

```

## Лабораторная работа 6

### Одномерные массивы

- 1 Составьте методы для ввода целочисленного одномерного массива из  $n$  элементов (возвращаемое значение – массив) и вывода массива.
- 2 Решите задачу в соответствии с вашим вариантом (без использования встроенных методов для массива).

- 3 Подготовьте ответы по следующим темам:
- Определение и назначение массива.
  - Описание и инициализация одномерного массива.
  - Использование цикла `foreach` при работе с массивом.

*Вариант 1*

- 1) Определить и вывести максимальное количество подряд идущих элементов, каждый из которых больше предыдущего.
- 2) Найти максимальный элемент массива, поставить его на среднее место в массиве, остальные элементы сдвинуть до освободившейся позиции.

*Вариант 2*

- 1) Найти и вывести два соседних элемента массива, значения которых наименее близки, то есть абсолютная величина их разности максимальна. Если таких пар несколько, то вывести первую из них.
- 2) Сдвинуть элементы массива влево на  $k$  позиций. Освободившиеся места заполнить нулями.

*Вариант 3*

- 1) Найти и вывести сумму элементов массива, стоящих после первого отрицательного числа.
- 2) Поставить все отрицательные элементы массива в его начало (порядок следования элементов может быть произвольным).

*Вариант 4*

- 1) Найти и вывести самую глубокую "яму" в массиве положительных чисел (значение которой минимально). Ямой называется не крайний элемент массива, который меньше обоих своих соседей. Например, в массиве из шести элементов, равных соответственно 4, 9, 2, 17, 3, 8, есть две ямы – 2 и 3, самая глубокая яма – 2.
- 2) Переписать элементы массива в обратном порядке.

*Вариант 5*

- 1) Найти и вывести среднее арифметическое тех элементов массива, которые по своему значению меньше последнего элемента этого массива.
- 2) Найти максимальный элемент массива, поставить его на последнее место. Остальные элементы сдвинуть до освободившейся позиции.

*Вариант 6*

- 1) Вычислить и вывести количество элементов массива, расположенных между минимальным и максимальным элементами.
- 2) Удалить из массива все отрицательные элементы с помощью сдвига остальных его элементов влево. Освободившиеся места заполнить нулями.

*Вариант 7*

- 1) Найти и вывести количество таких элементов массива, которые равны среднему арифметическому двух элементов, расположенных сразу после

него. Например, в массиве из 6 элементов, равных соответственно 2, 3, 1, 5, 6, 4, есть три таких элемента, они расположены на первом, втором и четвертом месте и равны 2, 3 и 5.

- 2) Изменить расположение элементов массива так, чтобы сначала в нем были нечетные числа, а затем – четные из первоначального массива.

#### *Вариант 8*

- 1) Определить есть ли в массиве хотя бы одна тройка соседних чисел, в которой средний элемент больше своих "соседей", т. е. предшествующего и последующего. В случае положительного ответа определить и вывести номера элементов первой из таких троек.
- 2) Сдвинуть элементы массива вправо на  $k$  позиций. Освободившиеся места заполнить нулями.

#### *Вариант 9*

- 1) Вычислить и вывести количество элементов массива, принадлежащих отрезку  $[a, b]$ .
- 2) Изменить элементы массива по следующему правилу: первый и последний элементы остаются без изменения, каждый из оставшихся элементов должен быть равен сумме собственного значения и соседних с ним элементов из первоначального массива.

#### *Вариант 10*

- 1) Найти и вывести сумму элементов самой длинной возрастающей последовательности подряд идущих элементов массива.
- 2) Даны два упорядоченных по возрастанию массива. Получить третий упорядоченный массив слиянием элементов из данных массивов.

#### *Вариант 11*

- 1) Найти и вывести количество пар элементов массива, в которых сумма элементов делится на 2, но не делится на 4. В данной задаче под парой подразумеваются два соседних элемента массива.
- 2) Поставить последний элемент массива на  $k$  место ( $k$  меньше длины массива). Остальные элементы сдвинуть до освободившейся позиции.

## **Лабораторная работа 7**

### **Сортировка одномерного массива**

- 1 Составьте методы для реализации алгоритмов сортировки (пузырьком, линейной, выбором и быстрой). Заполните целочисленный массив данными, полученными с помощью генератора случайных чисел. В методе Main() организуйте меню для выбора методов сортировки.
- 2 Протестируйте программу.
- 3 Задайте точки останова при обращении к каждому методу сортировки. Запустите программу в режиме отладки и, выполняя «шаги с заходом» (F11), проанализируйте последовательность действий (на вкладках «Локальные» и «Стек вызовов»).

- 4 Продемонстрируйте преподавателю результаты своей работы.
- 5 Подготовьте ответы на вопросы:
  - Что такое сортировка?
  - Что общего и чем отличаются различные алгоритмы сортировок?

## 5 Многомерные массивы

### 5.1 Двумерные массивы

Простейший многомерный массив – двумерный. В нем позиция любого элемента определяется двумя индексами. Если представить двумерный массив в виде таблицы данных или матрицы, то один индекс означает строку, а второй – столбец.

Описание двумерного массива с целочисленными значениями размером 10×20 с именем *table*:

```
int[,] table = new int[10, 20];
```

Синтаксис первой части этого объявления [,] означает, что создается ссылочная переменная двумерного массива. Для реального выделения памяти для этого массива с помощью оператора new используется более конкретный синтаксис: int[10, 20]

Чтобы получить доступ к элементу двумерного массива, необходимо указать оба индекса, разделив их запятой. Например, table[3, 5] = 10;

Рассмотрим пример программы, которая заполняет двумерный массив числами от 0 до 11 и выводит содержимое этого массива.

```
using System;
class TwoD
{
    public static void Main()
    {
        int k = 0;
        int[,] table = new int[3, 4];
        for (int i = 0; i < table.GetLength(0); i++)
        {
            for (int j = 0; j < table.GetLength(1); j++)
            {
                table[i, j] = k++;
                Console.Write(table[t, i] + " ");
            }
            Console.WriteLine();
        }
    }
}
```

В этом примере элемент массива table[0, 0] получит число 0, элемент table[0, 1] – число 1, элемент table[0, 2] – число 2 и т.д. Значение элемента table[2, 3] будет равно 11.

## 5.2 Массивы трех и более измерений

В C# можно определять массивы трех и более измерений. Объявление многомерного массива:

```
тип[, ... , ] имя = new тип[размер1, ..., размерN];
```

Например, с помощью следующего объявления создается трехмерный целочисленный массив:

```
int[, , ] multidim = new int[4, 10, 3];
```

```
multidim[2, 4, 1] = 100;
```

## 5.3 Массив массивов

Пример объявления ступенчатого массива:

```
//объявление массива, который содержит 3 массива  
int[][] array = new int[3][];  
//создание внутренних массивов  
array[0] = new int[3];  
array[1] = new int[2];  
array[2] = new int[5];
```

Доступ к элементам осуществляется по тому же принципу, как и с многомерными массивами, только тут уже участвуют две пары квадратных скобок:

```
array[0][1] = 5;  
array[1][1] = 8;  
array[1][2] = 5; // ошибка выполнения, индекс «2» вне границ массива
```

# Лабораторная работа 8

## Двумерные массивы

- 1 Решите задачи в соответствии с вашим вариантом.
- 2 Подготовьте ответы на следующие вопросы:
  - Как описывается матрица (двумерный массив) и выполняется обращение к ее элементам?
  - Каким образом элементы матрицы хранятся в оперативной памяти?
  - Как можно узнать размерности матрицы?

### Вариант 1

- 1) Ввести с консоли квадратную матрицу размерности  $n \times n$ . Вычислить сумму ее положительных элементов, расположенных по периметру и на диагоналях.
- 2) Ввести с консоли матрицу размерности  $m \times n$ . Поменять местами ее минимальный и максимальный элементы. Измененную матрицу вывести на консоль.

### Вариант 2

- 1) Ввести с консоли квадратную матрицу размерности  $n \times n$ . Проверить, является ли она единичной.
- 2) Ввести с консоли матрицу размерности  $m \times n$ . Поменять местами  $k$ -й и  $l$ -й столбцы матрицы ( $0 \leq k, l \leq n$ ). Измененную матрицу вывести на консоль.

### *Вариант 3*

- 1) Ввести с консоли квадратную матрицу размерности  $n \times n$ . Проверить, является ли она симметричной относительно главной диагонали.
- 2) Ввести с консоли матрицу размерности  $m \times n$ . Преобразовать исходную матрицу так, чтобы каждый элемент последнего столбца был заменен разностью максимального и минимального элементов строки, содержащей этот элемент. Измененную матрицу вывести на консоль.

### *Вариант 4*

- 1) Ввести с консоли матрицу размерности  $m \times n$ . Определить и вывести столбец с наименьшим количеством нечетных элементов (если таких столбцов несколько, то вывести столбец с наибольшим индексом), иначе вывести сообщение, что нечетных элементов в матрице нет.
- 2) Ввести с консоли матрицу размерности  $m \times n$ . Поменять в исходной матрице строки с четными и нечетными индексами ( $0 \leftrightarrow 1, 2 \leftrightarrow 3, \dots$ ). Измененную матрицу вывести на консоль.

### *Вариант 5*

- 1) Ввести с консоли матрицу размерности  $m \times n$ . Определить и вывести строку с наибольшим количеством отрицательных элементов (если таких строк несколько, то вывести строку с наименьшим индексом), иначе вывести сообщение, что отрицательных элементов в матрице нет.
- 2) Дана матрица размерности  $m \times n$ . Преобразовать исходную матрицу так, чтобы строки с четными индексами были упорядочены по возрастанию, а с нечетными – по убыванию. Измененную матрицу вывести на консоль.

### *Вариант 6*

- 1) Ввести с консоли квадратную матрицу размерности  $n \times n$ . Вычислить среднее арифметическое ее элементов, расположенных по периметру и на диагоналях.
- 2) Ввести с консоли матрицу размерности  $m \times n$ . В каждой строке исходной матрицы упорядочить элементы, расположенные между минимальным и максимальным элементами данной строки. Измененную матрицу вывести на консоль.

### *Вариант 7*

- 1) Ввести с консоли квадратную матрицу размерности  $n \times n$  ( $n > 2$ ). Определить номер столбца, в котором расположен минимальный элемент третьей строки матрицы. Вывести элементы этого столбца на консоль.
- 2) Дана матрица размерности  $m \times n$ . Из всех строк матрицы (кроме первой) вычесть первую строку, умноженную на элемент с индексом 0 из текущей строки. Измененную матрицу вывести на консоль.

### *Вариант 8*

- 1) Ввести с консоли матрицу размерности  $m \times n$ . Определить среднее арифметическое элементов матрицы, имеющих четные индексы строк и нечетные индексы столбцов.

- 2) Дана квадратная матрица размерности  $n \times n$ . Все симметричные относительно главной диагонали и равные элементы заменить нулями. Измененную матрицу вывести на консоль.

#### *Вариант 9*

- 1) Ввести с консоли матрицу размерности  $m \times n$ . Определить количество элементов матрицы, больших ее среднего значения.
- 2) Ввести с консоли матрицу размерности  $m \times n$ . Перед строкой, содержащей минимальный элемент матрицы (без учета нулевой строки) вставить нулевую строку, остальные строки сдвинуть. Измененную матрицу вывести на консоль.

#### *Вариант 10*

- 1) Дана квадратная матрица размерности  $n \times n$ . Вычислить сумму всех ее элементов за исключением диагональных элементов.
- 2) Ввести с консоли матрицу размерности  $m \times n$ . Строку, содержащую максимальный элемент матрицы, прибавить ко всем остальным строкам. Измененную матрицу вывести на консоль.

#### *Вариант 11*

- 1) Дана квадратная матрица размерности  $n \times n$ . Определить отдельно суммы ее нечетных и четных элементов.
- 2) Ввести с консоли матрицу размерности  $m \times n$ . Из всех нечетных элементов матрицы вычесть последний элемент соответствующего столбца. Измененную матрицу вывести на консоль.

## 6 Коллекции

Хотя в языке C# есть массивы, которые хранят в себе наборы однотипных объектов, но работать с ними не всегда удобно. Например, массив хранит фиксированное количество объектов. Однако если заранее не известно, сколько потребуется объектов, то в этом случае намного удобнее применять коллекции. Еще один плюс коллекций состоит в том, что некоторые из них реализуют стандартные структуры данных, например, стек, очередь, словарь, которые могут пригодиться для решения различных специальных задач.

Большая часть классов коллекций содержится в пространствах имен System.Collections (простые необобщенные классы коллекций), System.Collections.Generic (обобщенные или типизированные классы коллекций) и System.Collections.Specialized (специальные классы коллекций). Также для обеспечения параллельного выполнения задач и многопоточного доступа применяются классы коллекций из пространства имен System.Collections.Concurrent.

### 6.1 Класс List

Класс `List<T>` из пространства имен System.Collections.Generic представляет список однотипных объектов. Главное отличие от массива в том, что он динамический – можно вставлять и удалять элементы в любое время.



```

using System;
using System.Collections.Generic;

namespace Collections
{
    class Program
    {
        static void Main(string[] args)
        {
            // создание и инициализация списка
            List<string> teams = new List<string>() {"Динамо", "CSKA"};
            List<int> list = new List<int>(); // создание списка
            list.Add(0); // добавление элемента
            list.Add(2);
            list.Add(3);
            list.Insert(1, 1);
            list.RemoveAt(0);
            foreach (var e in list)
                Console.WriteLine(e);
        }
    }
}

```

Свойство *Count* определяет количество элементов, *Capacity* – зарезервированный объем памяти под список.

Стоит помнить, что простые массивы работают быстрее, чем списки *List*. Если в программе не особо важна производительность, то можно использовать список, иначе при неизменном количестве элементов нужно использовать массивы.

## 6.2 Класс String

Основным типом данных для хранения текстовой информации в языке C# является тип *string*. Этот тип данных является встроенным и соответствует классу *System.String*.

Для инициализации символьных строк используются *строковые литералы*, то есть последовательности символов, заключенные в двойные кавычки. Кроме символов строковый литерал может содержать *Escape*-последовательности – специальные символы, которые начинаются с обратного слеша (\), после которого следует обозначение символа.

Строка представляет собой массив символов, обратиться к каждому из которых можно с использованием индекса этого элемента.

Класс *string* содержит ряд конструкторов для инициализации строк, которые позволяют сформировать строку из массива символов типа *char* или его части. Также можно сформировать строку из последовательности одинаковых символов.

```

using System;
namespace MyProgram
{
    class Program

```

```

{
    static void Main(string[] args)
    {
        char[] m = { 'M', 'a', 'c', 'c', 'и', 'B' };
        string s1 = new string(m);
        string s2 = new string(m, 3, 2);
        string s3 = new string('A', 5);
        Console.WriteLine(s1);
        Console.WriteLine(s2);
        Console.WriteLine(s3);
        Console.ReadKey();
    }
}

```

Основным свойством любой строки является свойство *Length*, которое позволяет получить количество символов в строке.

Статический метод `Copy()` позволяет создать копию строки, указанной в качестве аргумента.

Метод `Compare()` является статическим методом класса `String` и позволяет посимвольно сравнить две строки или подстроки. Возвращаемое значение метода равно 0 в случае равенства строк. Пример использования этого метода:

Для проверки, содержат ли строки одинаковые последовательности символов, может использоваться метод `Equals()`, имеющий как статический, так и нестатический вариант использования. Пример использования методов:

```

using System;
namespace MyProgram
{
    class Program
    {
        static void Main(string[] args)
        {
            string s1 = "Привет";
            string s2;
            s2 = string.Copy(s1);
            Console.WriteLine(s1.Equals(s2));
            Console.WriteLine(string.Equals(s1, s2));
            s2 += "!";
            Console.WriteLine(s1);
            Console.WriteLine(s2);
            Console.ReadKey();
        }
    }
}

```

### 6.3 Словари

Еще один распространенный тип коллекции представляют словари. Словарь хранит объекты, которые представляют пару *ключ-значение*. Каждый такой объект является объектом структуры `KeyValuePair<TKey, TValue>`.

Свойства Key и Value позволяют получить ключ и значение элемента в словаре. Ключ элемента для словарей является его “индексом”.

Рассмотрим на примере использование словарей:

```
Dictionary<int, string> countries = new Dictionary<int, string>();
countries.Add(1, "Russia");
countries.Add(3, "Great Britain");
countries.Add(2, "USA");
countries.Add(4, "France");
countries.Add(5, "China");

foreach (KeyValuePair<int, string> item in countries)
{
    Console.WriteLine(item.Key + " - " + item.Value);
}

// получение элемента по ключу
string country = countries[4];
// изменение объекта
countries[4] = "Spain";
// удаление по ключу
countries.Remove(2);
// Инициализация словаря
Dictionary<string, string> countries = new Dictionary<string, string>
{
    {"Франция", "Париж"},
    {"Германия", "Берлин"},
    {"Великобритания", "Лондон"}
};

foreach (var pair in countries)
    Console.WriteLine("{0} - {1}", pair.Key, pair.Value);
Начиная с C# 6.0 доступен также еще один способ инициализации:
Dictionary<string, string> countries = new Dictionary<string, string>
{
    ["Франция"] = "Париж",
    ["Германия"] = "Берлин",
    ["Великобритания"] = "Лондон"
};
```

## Лабораторная работа 9

### Строки и словари

#### 1 Задачи на строки

- 1) Посчитать в строке количество символов ";" и ":". (3 баллов)
- 2) Дана строка. Вывести на экран ее символы, стоящие на четных местах (например, из строки "программа" вывести "ргам"). (3 баллов)
- 3) Даны два слова. Верно ли, что первое слово начинается на ту же букву, на которую заканчивается второе слово? (3 баллов)

- 4) Дано слово. Проверить, является ли оно перевертышем (одинаково читается слева направо и справа налево)? (5 баллов)
- 5) Дана строка из 12 символов. Поменять местами его трети следующим образом: третья треть, затем первая треть и на последнем месте – вторая треть. (5 баллов)
- 6) Удалить из строки все слова, начинающиеся на "о". (5 баллов)
- 7) Посчитать в строке количество слов, заканчивающихся на "ая". (6 баллов)
- 8) Удалить из строки все цифры и в конце каждого слова добавить символ ";". (6 баллов)
- 9) Текст, содержащий буквы (русского и английского алфавитов), пробелы и различные знаки препинания, ввести в переменную строкового типа. Определить количество слов в тексте. Найти и вывести самое короткое и длинное слово. (7 баллов)
- 10) Введена строка, содержащая данные о человеке: фамилию, имя и отчество. Регистр символов в ней может быть произвольным. Преобразовать эту строку таким образом, чтобы первые символы в ФИО были прописными, а остальные – строчными. (7 баллов)

## 2 Задачи на словари

- 1) Создать словарь по таблице о приросте населения за пять лет в городах (название города является ключом, а массив с данными о приросте населения – значением):

Город	Прирост населения, тыс. чел.				
	2011	2012	2013	2014	2015
Ижевск	2,5	1,3	-0,2	-0,1	0,6
...					

Определить и вывести средний прирост населения в каждом из введенных городов. (20 баллов)

- 2) Составить словарь «студентов-должников». Для каждого студента перечислить список дисциплин, по которым остались долги (ФИО студента – ключ словаря, список дисциплин – значения словаря). На основе этих данных получить другой словарь, в котором дисциплина является ключом, а список ФИО студентов – значениями. (30 баллов)
- 3) Для данных об абонентах (фамилия, имя и отчество каждого) и номерах телефонов создать словарь при условии, что у каждого абонента может быть любое количество телефонных номеров.
  - По введенному с консоли номеру телефона вывести фамилию и инициалы абонента.
  - По введенной фамилии вывести данные об абонентах с такой фамилией и все их телефонных номера. (40 баллов)

## 3 Ответить на вопросы (10 баллов):

- Как описывается строка, и каким образом она хранится в оперативной памяти?

- Какие действия можно выполнять над строками в программе? Какие изменения при этом выполняются в оперативной памяти?
- Что такое словарь? Как он описывается и заполняется в программе?
- Каким образом можно обратиться к его элементам (ключам и значениям)?
- Чем класс `StringBuilder` отличается от `String`? В каких случаях его следует использовать?

**Примечание.** Даны две группы задач с указанием баллов за каждую задачу. Из первой группы засчитываются все решенные задачи, из второй группы – только одна. После выполнения практических заданий нужно ответить на теоретические вопросы.

## 7 Организация системы ввода-вывода

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

На самом низком уровне иерархии потоков ввода-вывода находятся потоки, оперирующие байтами. Это объясняется тем, что многие устройства при выполнении операций ввода-вывода ориентированы на байты. Однако для человека привычнее оперировать символами, поэтому разработаны символьные потоки, которые фактически представляют собой оболочки, выполняющие преобразование байтовых потоков в символьные и наоборот. Кроме этого, реализованы потоки для работы с `int` -, `double` -, `short` – значениями, которые также представляют оболочку для байтовых потоков, но работают не с самими значениями, а с их внутренним представлением в виде двоичных кодов.

Центральную часть потоковой C#-системы занимает класс *Stream* пространства имен *System.IO*. Класс *Stream* представляет байтовый поток и является базовым для всех остальных потоковых классов. Из класса *Stream* выведены такие байтовые классы потоков как:

*FileStream* – байтовый поток, разработанный для файлового ввода-вывода

*BufferedStream* – включает в оболочку байтовый поток и добавляет буферизацию, которая во многих случаях увеличивает производительность программы;

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

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

## 7.1 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:

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

где:

- ✓ параметр *filename* определяет имя файла, с которым будет связан поток ввода-вывода данных; при этом *filename* определяет либо полный путь к файлу, либо имя файла, который находится в папке `bin/debug` проекта.
- ✓ параметр *mode* определяет режим открытия файла, который может принимать одно из возможных значений, определенных перечислением `FileMode`:

Если попытка открыть файл оказалась неуспешной, то генерируется исключение.

Другая версия конструктора позволяет ограничить доступ (только чтением или только записью) или не ограничивать его:

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

где:

параметр *how*, определяет способ доступа к файлу и может принимать одно из значений, определенных перечислением `FileAccess`.

*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 достигли конца файла, то метод `ReadByte()` возвращает значение `-1`.

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

Рассмотрим пример использования класса `FileStream`, для копирования одного файла в другой. Но вначале создадим текстовый файл `text1.txt` в папке `bin/debug` текущего проекта. И внесем в него произвольную информацию.

```
using System;
using System.IO; //для работы с потоками
namespace MyProgram
{
    class Program
    {
        static void Main()
        {
            try
            {
                FileStream fileIn = new FileStream("text1.txt",
                    FileMode.Open, FileAccess.Read);
                FileStream fileOut = new FileStream("text 2.txt",
```

```

        FileMode.Create, FileAccess.Write);
    int i;
    fileIn.CopyTo(fileOut);
    while ((i = fileIn.ReadByte()) != -1)
    {
        // Запись очередного байта в поток, связанный с файлом fileOut.
        fileOut.WriteByte((byte)i);
    }
    fileIn.Close();
    fileOut.Close();
}
catch (Exception ex)
{
    Console.WriteLine(ex.Message);
}
}
}
}

```

После завершения 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. Потоки содержат методы `Dispose()` и `Close()` (функционально идентичны, закрывают поток). Метод `Flush()` обеспечивает принудительную запись любых буферизированных данных.

Другой способ обеспечения закрытия потока после завершения работы – создание его экземпляров внутри блока *using*.

## 7.2 Символьный поток

`TextReader` и `TextWriter` являются абстрактными базовыми классами для адаптеров, которые имеют дело исключительно с символами и строками. Каждый из них имеет две реализации:

- ✓ `StreamReader` и `StreamWriter` – для хранения данных используют класс `Stream` и транслируют байты потока в символы или строки;
- ✓ `StringReader` и `StringWriter` – для хранения данных используют строки.

У символьных потоков есть свои преимущества. Например, можно использовать регулярные выражения для поиска заданных фрагментов текста в файле.

```

using System.Text.RegularExpressions;
//
static void Main()
{
    StreamReader fileIn = new StreamReader("text.txt");
    StreamWriter fileOut = new StreamWriter("newText.txt", false);
    string text = fileIn.ReadToEnd();
    Regex r = new Regex(@"[-+]?d+");
    Match integer = r.Match(text);
    while (integer.Success)
    {

```

```

        fileOut.WriteLine(integer);
        integer = integer.NextMatch();
    }
    fileIn.Close();
    fileOut.Close();
}

```

### 7.3 Двоичный поток

Файл – это упорядоченная и именованная последовательность байтов, имеющая постоянное хранилище. При работе с файлами используются пути к каталогам, запоминающие устройства, а также имена файлов и каталогов. В отличие от файла, поток – это последовательность байтов, которую можно использовать для записи или чтения из вспомогательного запоминающего устройства, являющегося одним из устройств хранения информации (например, дисков или памяти).

Двоичные файлы хранят данные в том же виде, в котором они представлены в оперативной памяти, то есть во внутреннем представлении. Двоичные файлы не применяются для просмотра человеком, они используются только для программной обработки.

Классы `BinaryReader` и `BinaryWriter` выполняют чтение и запись в поток predefined типов: `bool`, `byte`, `char`, `decimal`, `float`, `double`, `short`, `int`, `long`, `sbyte`, `ushort`, `uint` и `ulong`, а также строк и массивов predefined типов. В отличие от текстовых адаптеров двоичные адаптеры сохраняют predefined символы эффективнее. Основные методы входного потока `BinaryWriter`:

Пример формирования двоичного файла:

```

static void Main()
{
    // Открыть двоичный поток.
    BinaryWriter fOut = new BinaryWriter(new FileStream("file.dat",
        FileMode.Create));

    for (int i = 0; i <= 100; i += 2)
    {
        // Записать данные в двоичный поток.
        fOut.Write(i);
    }
    // Закрыть поток.
    fOut.Close();
}

```

Содержимое двоичного файла можно обрабатывать только программным путем, например, вывести его содержимое:

```

static void Main()
{
    FileStream f = new FileStream("t.dat", FileMode.Open);
    BinaryReader fIn = new BinaryReader(f);
    // Определить количество чисел в двоичном потоке.
    long n = f.Length / sizeof(Int32);
}

```



```

int a;
for (int i = 0; i < n; i++)
{
    a = fIn.ReadInt32();
    Console.Write(a + " ");
}
Console.ReadKey();
// Закрывать поток.
fIn.Close();
f.Close();
}

```

Класс `BinaryReader` может также выполнять чтение в байтовый массив:

```
byte[] data = new BinaryReader(s).ReadBytes ((int) s.Length);
```

При закрытии адаптера и потока без использования инструкции *using*, нужно всегда *сначала* закрывать или сбрасывать *адаптер*, и только *после этого* закрывать *поток*, иначе любые данные, буферизированные в адаптере, будут утеряны.

Адаптеры относятся к *необязательно* освобождаемым объектам. Это означает, что их *необязательно* закрывать перед закрытием потока. В большинстве случаев их достаточно просто сбросить. Это может быть удобно в ситуации, когда после завершения работы с адаптером.

#### 7.4 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:

```

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

```

Пример метода для перенаправления потока:

```

static void PrintArray(int[,] mas, string name)
{
    int n = mas.GetLength(0);
    int m = mas.GetLength(1);
    // Перенаправить стандартный выходной поток на fileOut.
    StreamWriter fileOut = new StreamWriter(name);
}

```

```

Console.SetOut(fileOut);
for (int i = 0; i < n; i++)
{
    for (int j = 0; j < m; j++) Console.Write("{0} ", mas[i, j]);
    Console.WriteLine();
}
fileOut.Close();
}

```

## 7.5 Работа с файловой системой

В пространстве имен **System.IO** предусмотрено четыре класса, которые предназначены для работы с файловой системой компьютера, т.е. для создания, удаления переноса и т.д. файлов и каталогов.

Два статических класса – **Directory** и **File**, реализуют свои возможности с помощью статических методов, поэтому классы используют без создания соответствующих объектов (экземпляров классов).

Классы **DirectoryInfo** и **FileInfo** обладают схожими функциональными возможностями (с **Directory** и **File** соответственно), но порождены от класса **FileSystemInfo** и поэтому реализуются путем создания соответствующих экземпляров классов.

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**.

Класс **File** определяет статические методы, позволяющие открыть файл и прочитать/записать его содержимое за один шаг:

**File.ReadAllText** – читает целый файл и возвращает его содержимое в виде одной строки;

**File.ReadAllLines** – читает целый файл и возвращает его содержимое в виде массива строк;

**File.ReadAllBytes** – читает целый файл и возвращает его содержимое в виде байтового массива;

**File.WriteAllText** – записывает строку в файл;

**File.WriteAllLines** – записывает массив строк в файл;

**File.WriteAllBytes** – записывает байтовый массив в файл;

**File.AppendAllText** – добавляет строку в конец файла;

## Лабораторная работа 10

### Файлы

#### 1 Задачи на текстовые файлы

- 1) Создать текстовый файл ("file1.txt"), открыть его в редакторе и ввести в него произвольный текст.
- 2) В программе открыть файл "file1.txt" на дозапись и дополнить его текстом, считанным с консоли. (10 баллов)
- 3) Вывести из файла "file1.txt" на консоль все строки, которые начинаются и заканчиваются одним и тем же символом. Если таких

строк в файле нет, то вывести соответствующее сообщение. (10 баллов)

- 4) Переписать в другой файл все строки из файла "file1.txt", вставив в начало каждой строки ее порядковый номер. (10 баллов)
- 5) Записать из файла "file1.txt" в новый файл первые 5 байт, затем байты с 15 по 30. (10 баллов)
- 6) Ввести с клавиатуры некоторую матрицу и число. Записать в текстовый файл результат умножения матрицы на число. (10 баллов)

## 2 Задачи на двоичные файлы

- 1) Создать программно двоичный файл, содержащий целые числа, полученные с помощью генератора случайных чисел. Вывести содержимое файла и закрыть его. (15 баллов)
- 2) Найти среднее арифметическое всех чисел этого файла. (10 баллов)
- 3) Создать файл, содержащий сведения о деталях (код детали, наименование (15 символов), количество). Вывести деталь по ее порядковому номеру в файле. (25 баллов)

## Лабораторная работа 11

### Работа с файловой системой

- 1 Создайте структуру каталогов **temp\prim** и **temp\second\total** в текущем каталоге.
- 2 В каталоге **temp** создайте несколько текстовых файла (количество запросите у пользователя). Перенаправьте консольный поток вывода поочередно в каждый из этих файлов. Введите в них произвольный текст с консоли.
- 3 Выведите на консоль список всех объектов, содержащихся в папке temp.
- 4 Измените даты создания (или изменения) нескольких файлов, дату последнего обращения к одному каталогу.
- 5 Создайте новый текстовый документ (объект класса *FileInfo*), описывающий «результаты работы».
- 6 Запишите в него данные обо всех каталогах и файлах по следующему плану:
- 7 Тип объекта; имя; полное имя; дата создания; день, месяц, год и время (по отдельности) последнего внесения изменений; содержимое (или сообщение, что содержимого нет).

## 8 Классы и объекты

Классы служат основанием для объектно-ориентированного программирования. В классе определяются данные и код, который выполняет действия над этими данными.

Код любой программы, написанной на C#, находится в пределах класса. Классы не только позволяют хранить в своем составе части кода, но они

представляют собой эффективное языковое средство для написания сложных программ.

Класс представляет собой шаблон, на основании которого определяется вид объекта. В нем указываются данные и код, который будет оперировать этим и данными.

Объекты – это экземпляры классов. Класс является логической абстракцией. Физическое представление класса в оперативной памяти возникнет лишь после того, как будет создан объект этого класса.

Правильно сконструированный класс должен определять *одну и только одну логическую сущность*. При написании кода действует правило: в одном кодовом файле (с расширением .cs) должен быть расположен только один класс (или его часть).

Все классы библиотеки .Net, а также все классы, которые создает программист в среде .Net, имеют одного общего предка – класс *Object*.

Описание класса содержит ключевое слово *class*, за которым следует его имя, а далее в фигурных скобках – тело класса. Кроме того, для класса можно задать его базовые классы (предки) и ряд необязательных атрибутов и спецификаторов, определяющих различные характеристики класса:

```
[ атрибуты ] [ спецификаторы ] class имя_класса [ : предки ]  
{тело_класса}
```

Простейший пример класса:

```
class Demo{ }
```

```
Demo a = new Demo (); // Создание экземпляра класса Demo
```

Для каждого объекта при его создании в памяти выделяется отдельная область, в которой хранятся его данные. Если достаточный для хранения объекта объем памяти выделить не удалось, то генерируется исключение *OutOfMemoryException*.

В общем случае класс может содержать следующие функциональные элементы:

- ✓ Данные: переменные или константы.
- ✓ Методы, реализующие не только вычисления, но и другие действия, выполняемые классом или его экземпляром.
- ✓ Свойства (определяют характеристики класса в соответствии со способами их задания и получения).
- ✓ Конструкторы (реализуют действия по инициализации экземпляров или класса в целом).
- ✓ Деструкторы (определяют действия, которые необходимо выполнить до того, как объект будет уничтожен).
- ✓ Типы (типы данных, внутренние по отношению к классу).
- ✓ Индексаторы (обеспечивают возможность доступа к элементам класса по их порядковому номеру).
- ✓ Операции (задают действия с объектами с помощью знаков операций).
- ✓ События (определяют уведомления, которые может генерировать класс).

Данные, содержащиеся в классе, могут быть переменными или константами. При описании данных также можно указывать атрибуты и

спецификаторы, задающие различные характеристики элементов. Синтаксис описания элемента данных:

[атрибуты] [спецификаторы] [const] тип имя [ = начальное\_значение ]

## Лабораторные работы 12-13

### Классы и объекты. Перегрузка методов и операций

- 1 Опишите класс «Почтовый адрес организации». Создать:
  - два конструктора для передачи объекту класса:
    - наименования организации;
    - наименования и адреса организации;
  - свойства класса для:
    - возвращения и установки (get; set) составных частей адреса;
    - возвращения наименования организации
  - методы для:
    - вывода данных;
    - определения, находится ли организация в интересующем городе.
- 2 Опишите класс «Комплексное число».
  - Создать индекатор, который по значению 0 возвращает действительную часть, по значению 1 – мнимую часть числа.
  - Перегрузить операции:
    - унарного минуса (задает комплексно сопряженное число);
    - сложения, вычитания, умножения и деления комплексных чисел.
  - Свойство для вычисления модуля комплексного числа.

## Лабораторная работа 14

### Наследование

- 1 Создать классы с полями, свойствами и методами:
  - базовый класс **Persona** (ФИО, дата рождения, адрес), позволяющий выводить на экран информацию о человеке, а также определять его возраст (на момент текущей даты);
  - производные классы для хранения данных о *студентах* (факультет, курс) и *преподавателях* (должность, стаж), со своими методами вывода информации на экран и определения соответствия или несоответствия объекта заданным критериям;
  - класс основной части программы, содержащий метод **Main()**, в котором создаются экземпляры классов и вызываются все их методы и свойства.
- 2 Выполнить задание в соответствии с вариантом:
  - в редакторе создать текстовый файл **input.txt** с исходной информацией о 10 (или более) объектах;
  - создать класс, содержащий поля, свойства и методы, необходимые для работы с объектом, реализовать для него метод **CompareTo**

интерфейса **IComparable**, перегрузить метод **ToString** базового класса **object** и необходимые операции отношения;

- в методе **Main()** для хранения данных организовать массив экземпляров класса;
- выполнить отбор и сортировку данных в соответствии с заданием;
- результирующую информацию записать в файл **output.txt**.

#### *Вариант 1*

Составить список студентов группы, включив следующие данные: ФИО, номер группы, результаты сдачи трех экзаменов. Вывести в новый файл информацию о студентах, сдавших сессию по всем предметам не менее чем на 50 баллов, отсортировав по номеру группы.

#### *Вариант 2*

Составить багажную ведомость камеры хранения, включив следующие данные: ФИО пассажира, количество вещей, общий вес вещей. Вывести в новый файл информацию о тех пассажирах, средний вес багажа которых превышает заданный, отсортировав их по количеству вещей, сданных в камеру хранения.

#### *Вариант 3*

Составить автомобильную ведомость, включив следующие данные: марка автомобиля, номер автомобиля, фамилия его владельца, год приобретения, пробег. Вывести в новый файл информацию об автомобилях, выпущенных ранее определенного года, отсортировав их по пробегу.

#### *Вариант 4*

Составить список сотрудников учреждения, включив следующие данные: ФИО, год принятия на работу, должность, зарплата, рабочий стаж. Вывести в новый файл информацию о сотрудниках, имеющих зарплату ниже определенного уровня, отсортировав их по рабочему стажу.

#### *Вариант 5*

Составить инвентарную ведомость склада, включив следующие данные: вид продукции, стоимость, сорт, количество. Вывести в новый файл информацию о той продукции, количество которой менее заданной величины, отсортировав ее по количеству продукции на складе.

#### *Вариант 6*

Составить инвентарную ведомость игрушек, включив следующие данные: название игрушки, ее стоимость (в руб.), возрастные границы детей, для которых предназначена игрушка. Вывести в новый файл информацию о тех игрушках, которые предназначены для детей от N до M лет, отсортировав их по стоимости.

#### *Вариант 7*

Составить список вкладчиков банка, включив следующие данные: ФИО, № счета, сумма, год открытия счета. Вывести в новый файл информацию о тех

вкладчиках, которые открыли вклад в текущем году, отсортировав их по сумме вклада.

#### *Вариант 8*

Составить список студентов, включающий фамилию, факультет, курс, группу, 5 оценок. Вывести в новый файл информацию о тех студентах, которые имеют хотя бы одну двойку, отсортировав их по курсу.

#### *Вариант 9*

Составить список студентов, включающий ФИО, курс, группу, результат забега. Вывести в новый файл информацию о студентах, показавших три лучших результата в забеге. Если окажется, что некоторые студенты получили такие же высокие результаты, то добавить их к списку победителей.

#### *Вариант 10*

Составить список студентов группы, включив следующие данные: ФИО, номер группы, результаты сдачи трех экзаменов. Вывести в новый файл информацию о студентах, сдавших сессию по всем предметам не менее чем на 50 баллов, отсортировав по номеру группы.

#### *Вариант 11*

Составить багажную ведомость камеры хранения, включив следующие данные: ФИО пассажира, количество вещей, общий вес вещей. Вывести в новый файл информацию о тех пассажирах, средний вес багажа которых превышает заданный, отсортировав их по количеству вещей, сданных в камеру хранения.

## **9 Задания для практических занятий и самостоятельной работы**

### 9.1 Типы данных и операции языка C#

1 Определить типы констант:

'a', '\n', '5'  
1, 123, -369  
20000000000  
12E5, 0.1e-5, .1e+2  
23l, -23L  
45u  
98ul, 0UL, 54Lu  
21.4f, -14.2e-21F  
7.5e28m  
0xFFFF, 0x10, 0x1f2a  
0b1, 0b1101  
"a", "line", "\n"

2 Определить значения переменных после выполнения операций:

```
int k = 4 * 5 / 3;  
var t = 5 / 2 + 0.5;
```

```
int k = 37 % 10;
int k = 37 % -10;
int k = -54 % 10;
int k = -54 % -10;
int x = 258; var i = (byte)x;
int m=(int)(0.29999*10);
float g==(int)(0.49999999f*10);
```

3 Упростить запись каждого из выражений:

```
i = i + 1; j = j - 1;
y = x + y; x = x + 1;
x = x + 1; y = y + x;
```

4 Определить значения переменных после выполнения операций, если

```
int x = 1, y = 2, z;
```

```
z = x = y;
```

```
z = x = ++y;
```

```
z = x = y++;
```

```
z = ++x + ++x;
```

```
z = x++ + x++;
```

```
z = x++ + ++x;
```

```
z = ++x + x++;
```

```
если int a, b, c, d;
```

```
b = d = - (a = c = 1);
```

5 Записать оператор присваивания с использованием тернарной операции для решения задач.

- Дано целое число  $x$ . Если оно положительное, то увеличить его на 1, иначе уменьшить на 1.
- Дано целое число  $x$ . Если оно четное, то увеличить его в 2 раза, иначе оставить без изменения.
- Даны три числа  $n, k, m$ . Вывести максимальное из этих чисел.

6 Определить значения выражений:

```
4 << 3
```

```
-4 << 3
```

```
4 >> 3
```

```
-4 >> 3
```

7 Используя битовую операцию  $\&$ , проверить число на четность-нечетность и вывести результат.

8 Определить и вывести, знак числа.

9 Составить программы для решения задач:

- Дано двузначное число. Найти сумму его цифр.
- Вычислите значение функции:  $y = |x-a| \cdot e^{2x}$ , где  $a = 2,5$  (константа).

## 9.2 Условные алгоритмы и рекурсия

1 Решить квадратное уравнение:  $ax^2 + bx + c = 0$ .



- 2 Вычислить стоимость покупки с учетом скидки. Если стоимость покупки до 1000 рублей (включительно) – скидка 5%, до 10000 рублей (включительно) – 10%, свыше 10000 рублей – 15%.
- 3 Составить рекурсивные функции для вычисления:
  - $n!$
  - $n!!$
- 4 Составить программу для нахождения:
  - НОД (наибольшего общего делителя) двух натуральных чисел;
  - НОК (наименьшего общего кратного) двух чисел;
  - НОД нескольких натуральных чисел.
- 5 Возвести число  $a$  в целую степень  $n$  ( $a^n$ ) без использования встроенной функции и цикла.
- 6 Вывести все целые числа из отрезка  $[k, n]$  и найти их сумму ( $k < n$ ).
- 7 Найти сумму цифр числа.
- 8 Найти сумму  $n$  слагаемых, вычисляемую по формуле:
  - $1 + \frac{1}{2} + \frac{1}{3} + \frac{1}{4} + \dots$
  - $1 + \frac{1}{2} + 3 + \frac{1}{4} + 5 + \dots$
  - $1 + a + a^2 + a^3 + a^4 + \dots$

### 9.3 Циклические алгоритмы

Дано число натуральное число  $n$ .

- 1 Получить и вывести  $n$  ( $n > 3$ ) первых элементов последовательности Фибоначчи (1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144, 233, 377, 610, ... – каждый элемент последовательности, начиная с третьего, равен сумме двух предыдущих элементов).
- 2 Дана последовательность из  $n$  целых чисел. Найти среднее арифметическое четных отрицательных элементов последовательности.
- 3 Последовательность целых чисел вводится на одной строке. Вывести нечетные числа из этой последовательности.
- 4 Дана последовательность из  $n$  целых чисел. Найти два элемента последовательности, которые меньше всех остальных.
- 5 Разложить число  $n$  ( $n > 1$ ) на простые множители.
- 6 Определить, является ли данное число простым.
- 7 Найти количество и сумму его цифр.
- 8 Вывести первую и последнюю цифры числа.
- 9 Получить новое число из цифр числа  $n$ , записанных в обратном порядке.

### 9.4 Одномерные массивы

- 1 Определить, сколько четных и сколько нечетных элементов содержится в одномерном массиве целых чисел.
- 2 Найти и вывести минимальный по абсолютной величине элемент массива с указанием его индекса.
- 3 Переставить элементы массива  $a[n]$  так, чтобы их начальные значения шли в следующем порядке:

$a_0 a_{n-2} a_2 a_{n-4} a_4 \dots a_{n-5} a_3 a_{n-3} a_1 a_{n-1}$ .

- 4 Сгруппировать элементы массива следующим образом: в начале массива все отрицательные, затем нулевые и в конце – все положительные элементы (с сохранением порядка следования элементов в каждой группе).
- 5 Написать программу, которая определяет, состоят ли два массива из одинаковых элементов (без учета порядка следования).
- 6 **Решето Эратосфена.** Дано натуральное число  $k > 2$ . Найти и вывести все простые число, не превосходящие  $k$ .
- 7 Дана квадратная матрица. Определить, является ли она магическим квадратом.

### 9.5 Матрицы – двумерные массивы

*Дана матрица размерности  $m \times n$ .*

- 1 Составить одномерный массив из максимальных элементов столбцов матрицы.
- 2 Поменять местами максимальный и минимальный элементы матрицы.
- 3 Поэлементно вычесть последний столбец из всех столбцов, кроме последнего.
- 4 Транспонировать данную матрицу.
- 5 Создать матрицу размерности  $m \times n$  из вещественных чисел. Поэлементно вычесть последнюю строку из всех строк, кроме последней.
- 6 Создать матрицу размерности  $m \times n$  из целых чисел. Транспонировать данную матрицу.

### 9.6 Коллекции

- 1 Списки:
  - Создать одномерный массив из  $n$  целых чисел, полученных с помощью генератора случайных чисел. Переписать из этого массива в список числа, принадлежащие отрезку  $[a, b]$ .
  - Ввести с клавиатуры текст. Все слова из этого текста, длина которых меньше  $n$  (где  $4 < n < 10$ ), записать в список. Значения из списка вывести на экран.
- 2 Словари:
  - Имеется набор данных, состоящий из пар положительных целых чисел. Для каждой пары чисел находится значение `pod` – наибольший общий делитель. Напишите эффективную по времени работы и по используемой памяти программу, которая будет определять, какое значение `pod` встречалось чаще всего. Если несколько значений `pod` встречалось одинаковое наибольшее количество раз, вывести их в порядке убывания.
  - Задача «Муравьи» из предыдущей лекции.

### 9.7 Файлы и потоки

- 1 Текстовые файлы:

- Дан текстовый файл. Переписать в новый файл все его строки, добавив в конец каждой строки количество символов в ней.
  - Дан текстовый файл. Найти и вывести самую короткую и самую длинную строки.
- 2 Двоичные файлы:
- Дана последовательность из  $n$  вещественных чисел. Записать все эти числа в двоичный файл. Вывести на экран все компоненты файла с четными номерами, меньшие заданного значения.
  - Записать в двоичный файл "таблицу значений" функции  $y = \sin^2 x$  на отрезке  $[a, b]$  с шагом  $dx$ . Вывести данные из файла на экран.
- 3 Перенаправление стандартных потоков:
- Текстовый файл "TextIn.txt" создать редактором. Ввести в него произвольный текст. Перенаправить консольный ввод на этот файл, а консольный вывод на новый текстовый файл "TextOut.txt". Переписать содержимое из "TextIn.txt" в "TextOut.txt" по одному слову на каждой строке (знаки препинания удалить).
  - Составить программу, в которой организовать считывание введенных с клавиатуры чисел (на строке вводить по два числа: целое и вещественное). Вывод чисел в соответствии с их типом перенаправить с консоли в два текстовых файла: "int.txt" и "double.txt".

## 9.8 Классы и объекты

- 1 Описать класс для вычисления и вывода значения выражения:  $\frac{a+b}{b \cdot c}$ .
- 2 Создать класс Point, разработав следующие элементы класса:
- Поля:  
int x, y;
  - Конструкторы, позволяющие создать экземпляр класса:  
с нулевыми координатами;  
с заданными координатами.
  - Методы, позволяющие:  
вывести координаты точки на экран;  
рассчитать расстояние от начала координат до точки;  
переместить точку на плоскости на вектор (a, b).
  - Свойства:  
получить-установить координаты точки (доступно для чтений и записи);  
позволяющие умножить координаты точки на скаляр (доступно только для записи).

## 9.9 Перегрузка методов и операций в классе

- 1 В класс **Point** (из прошлой лекции) добавить:
- Индексатор, позволяющий по индексу **0** обращаться к полю **x**, по индексу **1** – к полю **y**, при других значениях индекса формировать исключение (сообщение об ошибке);
  - перегрузку:

- операции ++ ( -- ): одновременно увеличивает (уменьшает) значение полей  $x$  и  $y$  на 1;
  - констант **true** и **false**: обращение к экземпляру класса дает значение **true**, если значение полей  $x$  и  $y$  совпадает, иначе **false**;
  - операции бинарный +: одновременно добавляет к полям  $x$  и  $y$  значение числа; складывает координаты двух точек;
  - преобразования типа **Point** в кортеж (и наоборот).
- 2 Создать класс для работы с двумерным массивом размерности  $m \times n$ .
- Организовать ввод и вывод элементов с помощью индекатора.
  - Определить для класса операции сложения и умножения (числа и массива; массива и числа; двух массивов, размеры которых позволяют выполнить данную операцию).

### 9.10 Наследование классов и интерфейсов

- 1 Организовать наследование классов. Создать:
- абстрактный класс **Figure** (фигура) с методами вычисления площади и периметра, а также методом, выводящим информацию о фигуре на экран;
  - производные классы: **Rectangle** (прямоугольник), **Circle** (круг), **Triangle** (треугольник) со своими методами вычисления площади и периметра;
  - массив  $n$  фигур и вывести полную информацию о фигурах на экран.
- 2 Использование типа данных – структура. Создать:
- структуру **Software** – Программное обеспечение (название, производитель, цена, дата установки, срок использования) с методами, позволяющими вывести на экран информацию о программном обеспечении, а также определить соответствие возможности использования (на момент текущей даты);
  - массив структур программного обеспечения, вывести полную информацию из него на экран, организовать поиск и вывод данных о программном обеспечении, которое допустимо использовать на текущую дату.

## 10 Темы для расчетного задания (для студентов очной формы обучения) или контрольной работы (для студентов заочной формы обучения)

Программные документы должны включать:

- текст программы, оформленный по ГОСТ 19.401;
- описание программы, выполненное по ГОСТ 19.402.

Текст программы должен быть написан на алгоритмическом языке C# и реализован в среде программирования Visual Studio.

Основные конструкции, операторы и правила создания программ на языке C# описаны в литературных источниках [1-6].

В программе, входящей в состав работы, необходимо в соответствии с полученным заданием организовать выполнение действий со свободным выбором пунктов меню, в которых предусмотреть следующие возможности:

- создание или дополнение данных в файл (в зависимости от типа файла и исходных данных);
- вывод содержимого исходного файла на экран;
- вывод в текстовый файл или на экран монитора данных (или списка) в зависимости от введенных критериев в запросе;
- выход из программы.

Кроме того, программа должна быть снабжена подсказками, необходимыми для работы пользователя, а текст программы должен содержать комментарии, описывающие назначение пользовательских классов, свойств, методов, переменных или отдельных частей программы.

Ниже приведены темы работы по вариантам и задания для их выполнения.

Условные обозначения: А – допускается использование только букв и символов-разделителей (точек, тире); 9 – допускается использование только цифр; X – возможно использование любых символов из кодовой таблицы Unicode.

*Вариант 1* Разработка программы «Списка экспортируемых товаров».

1 Редактором на диске создать файл со структурой:

Наименование товара	Страна экспорта	Цена, р.	Объем экспорта
X(20)	A(15)	9(5),9(2)	9(5)

- 2 Вывести в текстовый файл список стран, в которые экспортируется товар, объемом больше 50000.
- 3 Вывести на экран монитора видеogramмы, содержащие данные о товаре по запросу наименования товара; страны экспорта.

*Вариант 2* Разработка программы «Телефонный справочник».

1 Программно создать файл со структурой:

Ф.И.О. абонента	Адрес	Номер телефона	Год установки
A(15)	X(12)	X(8)	9(4)

- 2 Вывести в текстовый файл список абонентов, телефоны которых начинаются на введенную с клавиатуры цифру.
- 3 Вывести на экран монитора видеogramмы, содержащие данные об абоненте по запросу фамилии; адреса.

*Вариант 3* Разработка программы «Список машин автопарка».

1 Редактором на диске создать файл со структурой:

Марка машины	Номер машины	Год выпуска	Признак состояния
X(20)	X(6)	9(4)	9(1)

где признак состояния: 0 – свободна, 1 – в рейсе, 2 – на ремонте.

- 2 Вывести в текстовый файл список машин, находящихся в рейсе.
- 3 Вывести на экран монитора видеogramмы, содержащие данные о машине по запросу марки машины; номера.

*Вариант 4* Разработка программы «Список товаров в магазине».

1 Программно создать файл со структурой:

Наименование товара	Цена, р.	Количество	Признак дефицита
A(20)	9(8),9(2)	9(8)	9(1)

где признак дефицита: 0 – дефицитный, 1 – недефицитный.

2 Вывести в текстовый файл список дефицитных товаров.

3 Вывести на экран монитора видеогаммы, содержащие данные о товаре по запросу наименования; цены.

*Вариант 5* Разработка программы «Расписание поездов».

1 Редактором на диске создать файл со структурой:

Номер поезда	Рейс	Время прибытия	Стоянка, мин.
9(3)	A(30)	X(5)	9(2)

2 Вывести в текстовый файл список поездов, время стоянки которых больше 30 минут.

3 Вывести на экран монитора видеогаммы, содержащие данные о поезде по запросу номера; времени прибытия.

*Вариант 6* Разработка программы «Список товаров в магазине игрушек».

1 Программно создать файл со структурой:

Название игрушки	Цена, р.	Возрастные границы, лет	
		от	до
A(20)	9(5),9(2)	9(2)	9(2)

2 Вывести в текстовый файл список игрушек, цена которых не превышает 500 рублей, предназначенных для детей 5 лет.

3 Вывести на экран монитора видеогаммы, содержащие информацию о самой дорогой игрушке в магазине и данные об игрушках по введенному названию.

*Вариант 7* Разработка программы «Библиотечный каталог».

1 Редактором на диске создать файл со структурой:

Фамилия автора	Название книги	Жанр	Год издания
A(15)	X(25)	A(10)	9(4)

2 Вывести в текстовый файл список книг, изданных после введенного в запросе года.

3 Вывести на экран монитора видеогаммы, содержащие список книг по запросу фамилии автора; жанра.

*Вариант 8* Разработка программы «Учет оплаты коммунальных услуг».

1 Программно создать файл со структурой:

Фамилия собственника	Адрес	Оплата, р.		
		за квартиру	за эл.энергию	за телефон
A(20)	X(15)	9(6), 9(2)	9(6), 9(2)	9(6), 9(2)

2 Вывести в текстовый файл список задолжников по оплате:

- за квартиру;
- за телефон.

- 3 Вывести на экран монитора видеogramмы, содержащие данные об оплате по запросу фамилии; адреса.

*Вариант 9* Разработка программы «Список подписных изданий».

- 1 Редактором на диске создать файл со структурой:

Тип издания	Название	Количество подписчиков	Цена, р.
A(10)	X(20)	9(4)	9(6), 9(2)

- 2 Вывести в текстовый файл список газет, на которые подписались более 100 человек.
- 3 Вывести на экран монитора видеogramмы, содержащие данные о подписном издании по запросу названия; цены.

*Вариант 10* Разработка программы «Регистрация участников конференции».

- 1 Программно создать файл со структурой:

Номер школы	Город	Ф.И.О. учителя	Название секции
9(4)	A(15)	A(20)	A(15)

- 2 Вывести в текстовый файл состав делегации по введенному в запросе городу.
- 3 Вывести на экран монитора видеogramмы, содержащие данные об учителе по запросу фамилии; список учителей по введенному названию секции.

*Вариант 11* Разработка программы «Список воспитанников детского сада».

- 1 Редактором на диске создать файл со структурой:

Ф.И.О. ребёнка	Номер группы	Признак содержания	Сумма оплаты, р.
A(15)	9(3)	9(1)	9(6), 9(2)

где признак состояния: 0 – гос. обеспечение, 1 – оплата родителей.

- 2 Вывести в текстовый файл список детей, находящихся на государственном обеспечении.
- 3 Вывести на экран монитора видеogramмы, содержащие данные о детях по запросу фамилии; номера группы.

*Вариант 12* Разработка программы «Учет грузоперевозок».

- 1 Программно создать файл со структурой:

Номер рейса	Номер квитанции	Вес багажа, кг.	Адрес доставки
X(6)	9(3)	9(6), 9(3)	X(25)

- 2 Вывести в текстовый файл данные о грузоперевозках с весом багажа свыше заданного количества.
- 3 Вывести на экран монитора видеogramмы, содержащие данные о грузоперевозках по запросу номера рейса; адреса доставки.

*Вариант 13* Разработка программы «Список проданных автомобилей».

- 1 Редактором на диске создать файл со структурой:

Марка автомобиля	Номер	Стоимость, р.	Ф.И.О. владельца
X(14)	X(6)	9(10), 9(2)	X(22)

- 2 Вывести в текстовый файл список автомобилей, стоимость которых больше 500000 рублей.

- 3 Вывести на экран монитора видеogramмы, содержащие данные об автомобилях по запросу марки; фамилии владельца.

*Вариант 14* Разработка программы «Список материалов на складе».

- 1 Программно создать файл со структурой:

Шифр материала	Наименование материала	Дата поступления	Количество, кг.
9(4)	X(12)	X(10)	9(4)

- 2 Вывести в текстовый файл список материалов, количество которых превышает введенное в запросе значение.
- 3 Вывести на экран монитора видеogramмы, содержащие данные о материале по запросу шифра; даты поступления.

*Вариант 15* Разработка программы «Учет рабочего времени».

- 1 Редактором на диске создать файл со структурой:

Табельный номер	Ф.И.О. сотрудника	Дата	Признак
9(6)	A(16)	X(10)	X(1)

где признак: 8 – выход на смену, Б – невыход по болезни, О – отпуск.

- 2 Вывести в текстовый файл список неработающих по болезни.
- 3 Вывести на экран монитора видеogramмы, содержащие данные о сотруднике по запросу табельного номера; список сотрудников, находящихся на работе по запросу даты.

*Вариант 16* Разработка программы «Список сотрудников предприятия».

- 1 Программно создать файл со структурой:

Табельный номер	Ф.И.О. сотрудника	Номер цеха	Должность
9(6)	A(17)	9(3)	A(15)

- 2 Вывести в текстовый файл список сотрудников по запросу номера цеха.
- 3 Вывести на экран монитора видеogramмы, содержащие данные о сотрудниках по запросу табельного номера; должности.

*Вариант 17* Разработка программы «Учет заработной платы».

- 1 Редактором на диске создать файл со структурой:

Ф.И.О. сотрудника	Лицевой счёт	Номер цеха	Заработная плата, р.
A(15)	9(8)	9(3)	9(6),9(2)

- 2 Вывести в текстовый файл список по запросу номера цеха и общую сумму выплат.
- 3 Вывести на экран монитора видеogramмы, содержащие данные о сотрудниках по запросу фамилии; лицевого счета.

*Вариант 18* Разработка программы «Учет данных о сотрудниках».

- 1 Программно создать файл со структурой:

Ф.И.О. сотрудника	Адрес	Год рождения	Семейное положение
A(15)	X(25)	9(4)	9(1)

где семейное положение: 0 – холост, 1 – женат (замужем).

- 2 Вывести в текстовый файл список холостых сотрудников.



- 3 Вывести на экран монитора видеогаммы, содержащие данные о сотрудниках по запросу фамилии; года рождения.

*Вариант 19* Разработка программы «Список фильмов в кинотеатрах».

- 1 Редактором на диске создать файл со структурой:

Название кинотеатра	Название фильма	Возрастные ограничения, лет	Время сеанса
A(13)	X(25)	9(2)	X(5)

- 2 Вывести в текстовый файл список вечерних сеансов (после 18.00).  
3 Вывести на экран монитора видеогаммы, содержащие списки фильмов по запросу кинотеатра; возрастных ограничений.

*Вариант 20* Разработка программы «Учет проданных билетов».

- 1 Программно создать файл со структурой:

Номер поезда	Станция назначения	Время отправления	Цена билета, р.	Количество билетов
9(3)	A(16)	X(5)	9(6), 9(2)	9(3)

- 2 Вывести в текстовый файл список поездов с ценой билета свыше 1000 рублей.  
3 Вывести на экран монитора видеогаммы, содержащие данные о проданных билетах по запросу номера поезда; станции назначения.

*Вариант 21* Разработка программы «Список врачей».

- 1 Редактором на диске создать файл со структурой:

Ф.И.О. врача	Специальность	Категория	Номер кабинета
A(15)	A(10)	9(1)	9(3)

- 2 Вывести в текстовый файл список стоматологов.  
3 Вывести на экран монитора видеогаммы, содержащие данные о врачах по запросу фамилии; список врачей по запросу категории.

*Вариант 22* Разработка программы «Каталог музея».

- 1 Программно создать файл со структурой:

Инвентарный номер	Наименование произведения	Автор работы	Год издания	Стоимость, р.
X(10)	A(20)	A(15)	9(4)	9(6),9(2)

- 2 Вывести в текстовый файл список произведений, изготовленных до 2000 года.  
3 Вывести на экран монитора видеогаммы, содержащие данные о произведениях по запросу автора работы; стоимости.

*Вариант 23* Разработка программы «Список жильцов».

- 1 Редактором на диске создать файл со структурой:

Ф.И.О. собственника	Адрес			Количество проживающих
	Улица	Дом	Квартира	
A(20)	X(15)	X(5)	9(3)	9(2)

- 2 Вывести в текстовый файл список жильцов по запросу улицы и дома.

- 3 Вывести на экран монитора видеogramмы, содержащие данные о собственнике квартиры по запросу фамилии; список собственников по запросу количества проживающих.

*Вариант 24* Разработка программы «Список фирм, производящих ЭВМ».

- 1 Программно создать файл со структурой:

Наименование фирмы	Адрес фирмы	Марка ЭВМ	Стоимость комплекта, р.
A(12)	X(20)	X(10)	9(8), 9(2)

- 2 Вывести в текстовый файл данные о ПЭВМ по запросу наименования фирмы.
- 3 Вывести на экран монитора видеogramмы, содержащие информацию о самом дешёвом комплекте; список ЭВМ по запросу стоимости.

*Вариант 25* Разработка программы «Список товаров».

- 1 Редактором на диске создать файл со структурой:

Наименование товара	Артикул	Количество	Цена, р.
A(20)	X(8)	9(4)	9(8), 9(2)

- 2 Вывести в текстовый файл список товаров, стоимость которых больше заданного значения.
- 3 Вывести на экран монитора видеogramмы, содержащие данные о товарах по запросу наименования; количества.

*Вариант 26* Разработка программы «Список хоккеистов».

- 1 Программно создать файл со структурой:

Ф.И.О. хоккеиста	Название команды	Заброшено шайб	Голевых передач	Штрафов
A(18)	X(10)	9(2)	9(2)	9(2)

- 2 Вывести в текстовый файл список хоккеистов, у которых нет штрафов.
- 3 Вывести на экран монитора видеogramмы, содержащие данные о хоккеисте по запросу фамилии; список хоккеистов по запросу названия команды.

*Вариант 27* Разработка программы «Список городов».

- 1 Редактором на диске создать файл со структурой:

Название города	Область (край)	Год основания	Число жителей
A(20)	A(20)	9(4)	9(8)

- 2 Вывести в текстовый файл список городов, основанных более 200 лет назад.
- 3 Вывести на экран монитора видеogramмы, содержащие списки городов по запросу названия области (края); числа жителей.

*Вариант 28* Разработка программы «Списка вузов».

- 1 Программно создать файл со структурой:

Название вуза	Город	Год аккредитации	Количество студентов
A(7)	A(14)	9(4)	9(4)

- 2 Вывести в текстовый файл список вузов по запросу города. вывести общее количество студентов в этом городе.
- 3 Вывести на экран монитора видеодиаграммы, содержащие данные о вузе по запросу названия; список вузов по запросу года аккредитации.

*Вариант 29* Разработка программы «Список горных вершин».

- 1 Редактором на диске создать файл со структурой:

Название вершины	Высота, м.	Государство	Признак
A(15)	9(4)	A(14)	9(1)

где признак: 0 – недоступен для туристов; 1 – платный доступ; 2 – свободный доступ.

- 2 Вывести в текстовый файл список гор, доступных для туристов.
- 3 Вывести на экран монитора видеодиаграммы, содержащие списки горных вершин по запросу названия государства; список государств, имеющих горы высотой больше заданного значения.

*Вариант 30* Разработка программы «Список стран».

- 1 Программно создать файл со структурой:

Название страны	Столица	Население страны	Государственный язык
A(15)	A(12)	9(8)	A(15)

- 2 Вывести в текстовый файл список самых населённых стран.
- 3 Вывести на экран монитора видеодиаграммы, содержащие данные о стране по запросу названия; список стран по запросу государственного языка.

## Список использованной литературы

- 1 Котов, О.М. Язык C#: краткое описание и введение в технологии программирования : учебное пособие / О.М. Котов ; Уральский федеральный университет им. первого Президента России Б. Н. Ельцина. – Екатеринбург : Издательство Уральского университета, 2014. – 209 с. : ил., табл., схем. – Режим доступа: по подписке. – URL: <https://biblioclub.ru/index.php?page=book&id=275809> (дата обращения: 24.01.2021). – Библиогр. в кн. – ISBN 978-5-7996-1094-4. – Текст : электронный.
- 2 Суханов, М.В. Основы Microsoft .NET Framework и языка программирования C# : учебное пособие / М.В. Суханов, И.В. Бачурин, И.С. Майоров ; Северный (Арктический) федеральный университет им. М. В. Ломоносова. – Архангельск : Северный (Арктический) федеральный университет (САФУ), 2014. – 97 с. : схем., табл., ил. – Режим доступа: по подписке. – URL: <https://biblioclub.ru/index.php?page=book&id=312313> (дата обращения: 24.01.2021). – Библиогр. в кн. – ISBN 978-5-261-00934-4. – Текст : электронный.
- 3 Горелов, С.В. Современные технологии программирования: разработка Windows-приложений на языке C#: учебник для студентов, обучающихся по дисциплине «Современные технологии программирования», направление «Прикладная информатика» (09.03.03 — для бакалавров, 09.04.03 — для магистров) : в 2 томах : [16+] / С.В. Горелов ; под науч. ред. П.Б. Лукьянова ; Финансовый университет при Правительстве Российской Федерации. – Москва : Прометей, 2019. – Том 1. – 363 с. : ил. – Режим доступа: по подписке. – URL: <https://biblioclub.ru/index.php?page=book&id=576037> (дата обращения: 24.01.2021). – Библиогр. в кн. – ISBN 978-5-907100-09-1. – Текст : электронный.
- 4 Горелов, С.В. Современные технологии программирования: разработка Windows-приложений на языке C#: учебник для студентов, обучающихся по дисциплине «Современные технологии программирования», направление «Прикладная информатика» (09.03.03 — для бакалавров, 09.04.03 — для магистров) : в 2 томах : [16+] / С.В. Горелов ; под науч. ред. П.Б. Лукьянова ; Финансовый университет при Правительстве Российской Федерации. – Москва : Прометей, 2019. – Том 2. – 379 с. : ил. – Режим доступа: по подписке. – URL: <https://biblioclub.ru/index.php?page=book&id=576036> (дата обращения: 24.01.2021). – Библиогр. в кн. – ISBN 978-5-907100-18-3. – Текст : электронный.
- 5 Документация по C# [режим доступа] <https://docs.microsoft.com/ru-ru/dotnet/csharp/>
- 6 Программирование на C# и .NET [режим доступа] <https://metanit.com/sharp/>