Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение
высшего образования
«Московский государственный технический университет имени Н.Э. Баумана
(национальный исследовательский университет)»
(МГТУ им. Н.Э. Баумана)

ФАКУЛЬТЕТ **ИНФОРМАТИКА И СИСТЕМЫ УПРАВЛЕНИЯ**

КАФЕДРА **КОМПЬЮТЕРНЫЕ СИСТЕМЫ И СЕТИ (ИУ6)**

Методические к лабораторным работам по дисциплине “Разработка приложений на языке C#”

2022 г.

СОДЕРЖАНИЕ

Лабораторная работа №1. Основы языка C#

Теоретическая часть:

Microsoft® .NET – это платформа нового поколения для создания распределенных бизнес-приложений. Основа .Net – это Microsoft .Net Framework – своеобразный верстак, набор средств и технологий по разработке и выполнению таких приложений. .NET имеет в себе общеязыковую среду времени выполнения(CLR, common language runtime). Любой программный код, написанный под новую платформу называется управляемым (managed code) и компилируется в бинарный вид, понятный .NET runtime. Этот формат называется Microsoft Intermediate Language (MSIL, IL). В связи с этим появляется возможность легкой интеграции кодов, написанных на разных языках программирования, т.к. платформа более низкого уровня у них одна.

В среду включен автоматический подсчет ссылок и сборщик мусора. Среда предоставляет расширенные возможности по управлению безопасностью кода, имеет стандартные возможности по работе с различными версиями одних и тех же компонент (versioning).

В платформу .Net входят много языков, которые без особого труда интегрируются друг с другом. В данных ЛР будет рассматриваться язык C#, так как именно этот язык был создан специально для данной платформы, прочие языки были лишь адаптированы для данной платформы. Язык C# схож с языками C++ и Java, но, безусловно, имеет и отличия.

Пример программы на C#:

```
class TestClass
{
    static void Main(string[] args)
    {
        // Display the number of command line arguments.
        Console.WriteLine(args.Length);
    }
}
```

Метод Main — это точка входа приложения C#. (Библиотекам и службам точка входа в виде метода Main не требуется.) Когда приложение запускается, первым вызывается именно метод Main.

В программе на C# может существовать только одна точка входа. Если у вас есть несколько классов с методом Main, программу нужно компилировать с параметром компилятора **StartupObject**, чтобы указать, какой из методов Main будет использоваться в качестве точки входа.

Если вы включаете директивы using, они должны быть первыми в файле.

Система типов C#:

C# является строго типизированным языком. Каждая переменная и константа имеет тип, как и каждое выражение, результатом вычисления которого является значение. Каждое объявление метода задает имя, тип и вид (значение, ссылка или вывод) для каждого входного параметра и для возвращаемого значения.

В библиотеке классов .NET определены встроенные числовые типы и комплексные типы, представляющие разнообразные конструкции. К ним относятся файловая система, сетевые подключения, коллекции и массивы объектов, а также даты. Обычная программа на C# использует типы из этой библиотеки классов и пользовательские типы, которые моделируют уникальные концепции конкретной сферы применения.

Когда вы объявляете в программе переменную или константу, для нее нужно задать тип либо использовать ключевое слово *var*, чтобы компилятор определил тип самостоятельно. В следующем примере показаны некоторые объявления переменных, использующие встроенные числовые типы и сложные пользовательские типы:

```
// Declaration only:
float temperature;
string name;
MyClass myClass;

// Declaration with initializers (four examples):
char firstLetter = 'C';
var limit = 3;
int[] source = { 0, 1, 2, 3, 4, 5 };
var query = from item in source
            where item <= limit
            select item;
```

C# предоставляет стандартный набор встроенных типов. Они используются для представления целых чисел, значений с плавающей запятой, логических выражений, текстовых символов, десятичных значений и других типов данных. Также существуют встроенные типы string и object.

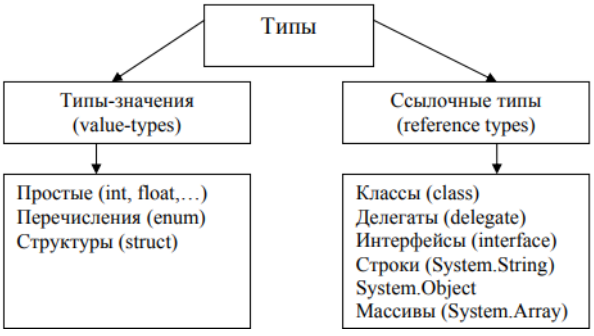

Рисунок 1 – Типы в C#

Действия программы выражаются с помощью *операторов*. C# поддерживает несколько типов операторов, некоторые из которых определяются как внедренные операторы.

- С помощью *блоков* можно использовать несколько операторов в таких контекстах, где ожидается только один оператор. Блок состоит из списка инструкций, заключенных между разделителями { и }.
- *Операторы объявления* используются для объявления локальных переменных и констант.
- *Операторы выражений* позволяют вычислять выражения. В качестве оператора можно использовать такие выражения, как вызовы методов, выделение объектов с помощью оператора new, назначения спомощью = и составных операторов присваивания, операторы ++ и -
  - для приращения и уменьшения, а также выражения await.
- *Операторы выбора* используются для выбора одного оператора из нескольких возможных вариантов в зависимости от значения какого-либо выражения. Эта группа содержит операторы if и switch.
- *Операторы итерации* используются для многократного выполнения внедренного оператора. Эта группа содержит операторы while, do, for и foreach.
- *Операторы перехода* используются для передачи управления. Эта группа содержит операторы break, continue, goto, throw, return и yield.
- Операторы try...catch позволяют перехватывать исключения, создаваемые при выполнении блока кода, а оператор try...finally используется для указания кода завершения, который выполняется всегда, независимо от появления исключений.
- Операторы checked и unchecked операторы позволяют управлять контекстом проверки переполнения для целочисленных арифметических операций и преобразований.
- Оператор lock позволяет создать взаимоисключающую блокировку заданного объекта перед выполнением определенных операторов, а затем снять блокировку.
- Оператор using используется для получения ресурса перед определенным оператором, и для удаления ресурса после его завершения.

Задание:

Создать консольное приложение – в соответствии с вариантом.

Варианты:

N Варианта	Задание
1	Создать консольное приложение, которое просит пользователя

	ввести два числа и вычисляет их сумму, разность, произведение и частное.
2	Создать консольное приложение, которое запрашивает у пользователя длины трех сторон треугольника и определяет, является ли он прямоугольным.
3	Создать консольное приложение, которое просит пользователя ввести целое число и выводит все его делители.
4	Создать консольное приложение, которое принимает от пользователя строку и определяет, является ли она палиндромом, игнорируя пробелы и знаки препинания.
5	Создать консольное приложение, которое генерирует случайный пароль заданной длины, включая цифры, буквы верхнего и нижнего регистра, а также специальные символы.
6	Создать консольное приложение, которое принимает от пользователя число и вычисляет его факториал, используя рекурсию.
7	Создать консольное приложение, которое просит пользователя ввести строку, а затем находит и выводит все слова, начинающиеся с буквы "C".
8	Создать консольное приложение, которое принимает от пользователя дату в формате "год-месяц-день" и определяет, является ли она валидной датой.
9	Создать консольное приложение, которое просит пользователя ввести числа в одной строке, разделенные пробелами, и сортирует их в порядке убывания.
10	Создать консольное приложение, которое принимает от пользователя число и выводит на экран все простые числа до этого числа.
11	Создать консольное приложение, которое просит пользователя ввести строку, а затем определяет, сколько раз встречается в ней заданное слово.
12	Создать консольное приложение, которое принимает от пользователя длину массива и элементы массива, а затем находит среднее арифметическое и медиану этого массива.
13	Создать консольное приложение, которое просит пользователя ввести число и находит все его простые делители.
14	Создать консольное приложение, которое принимает от пользователя две строки и определяет, является ли одна из них анаграммой другой.
15	Создать консольное приложение, которое просит пользователя ввести строку и находит и выводит самое длинное слово в ней.
16	Создать консольное приложение, которое принимает от пользователя число и проверяет, является ли оно числом

	Фибоначчи.
17	Создать консольное приложение, которое просит пользователя ввести число и выводит на экран его бинарное представление.
18	Создать консольное приложение, которое принимает от пользователя строку и определяет, является ли она панграммой (содержит все буквы алфавита).
19	Создать консольное приложение, которое просит пользователя ввести дату рождения и выводит на экран его возраст.
20	Создать консольное приложение, которое принимает от пользователя число и проверяет, является ли оно совершенным числом (сумма всех делителей равна самому числу).
21	Создать консольное приложение, которое просит пользователя ввести две даты и вычисляет разницу между ними в днях.
22	Создать консольное приложение, которое принимает от пользователя строку и определяет, является ли она палиндромом, игнорируя регистр букв.
23	Создать консольное приложение, которое просит пользователя ввести набор чисел и определяет, сколько из них являются простыми.
24	Создать консольное приложение, которое принимает от пользователя строку и определяет, сколько раз в ней встречается заданная подстрока.
25	Создать консольное приложение, которое просит пользователя ввести число и выводит на экран его прописную форму (например, для числа 42 - "сорок два").
26	Создать консольное приложение, которое принимает от пользователя строку и определяет, является ли она панграммой (содержит все буквы алфавита, как минимум один раз), игнорируя регистр букв и пробелы.
27	Создать консольное приложение, которое просит пользователя ввести число и определяет, является ли оно числом Армстронга (число, равное сумме своих цифр, возведенных в степень количества цифр в числе).
28	Создать консольное приложение, которое принимает от пользователя строку и находит и выводит на экран самую длинную подстроку без повторяющихся символов.
29	Создать консольное приложение, которое просит пользователя ввести число и выводит на экран его простой множитель.
30	Создать консольное приложение, которое принимает от пользователя строку и определяет, является ли она палиндромом, если игнорировать пробелы, знаки препинания и регистр букв.

Структура отчета по лабораторной работе:

Отчет является документом, свидетельствующим о выполнении студентом лабораторной работы. Отчет должен включать:

1) Титульный лист с номером и названием лабораторной группы, идентификатором группы, номером варианта, фамилией студента, датой сдачи отчета по домашнему заданию, местом для подписи преподавателя;
2) Введение, цель и задачи лабораторной работы;
3) Описание работы программы;
4) Схему алгоритма;
5) Листинг программы (код должен быть самодокументирован, т.е. содержать комментарии);
6) Демонстрацию работы приложения (скриншоты ручного тестирования)
7) Вывод

Контрольные вопросы:

1) Какие типы данных существуют в C#?
2) Какие типы операторов вам известны?
3) Что является точкой входа в приложения, написанные на C#?
4) Какая разница между циклами for и while? Когда вы бы предпочли использовать один из них перед другим?
5) Каким образом можно получить ввод от пользователя через консольный интерфейс?

Лабораторная работа №2. Массивы

Теоретическая часть:

Массив — это структура данных, содержащая ряд переменных, к которым осуществляется доступ с помощью вычисляемых индексов. Все содержащиеся в массиве переменные, также называемые *элементами массива*, относятся к одному типу. Этот тип называется *типом элемента* массива.

Сами массивы имеют ссылочный тип, и объявление переменной массива только выделяет память для ссылки на экземпляр массива. Фактические экземпляры массива создаются динамически во время выполнения с помощью оператора new.

Операция new задает *длину* нового экземпляра массива, который затем фиксируется на время существования экземпляра. Элементы массива имеют индексы в диапазоне от 0 до Length - 1. Оператор new автоматически инициализирует все элементы массива значением по умолчанию. Например, для всех числовых типов устанавливается нулевое значение, а для всех ссылочных типов — значение null.

Следующий пример кода создает массив из int элементов, затем инициализирует этот массив и выводит содержимое массива.

```
int[] a = new int[10];
for (int i = 0; i < a.Length; i++)
{
    a[i] = i * i;
}
for (int i = 0; i < a.Length; i++)
{
    Console.WriteLine($"a[{i}] = {a[i]}");
}
```

Кроме этого, C# поддерживает *многомерные массивы*. Число измерений типа массива, также известное как *ранг* типа массива, равно одному плюс числу запятых между квадратными скобками типа массива. Следующий пример кода поочередно создает одномерный, двухмерный и трехмерный массивы.

```
int[] a1 = new int[10];
int[,] a2 = new int[10, 5];
int[,,] a3 = new int[10, 5, 2];
```

Элементы массива могут иметь любой тип, в том числе тип массива. Следующий пример создает массив массивов int.

```
int[][] a = new int[3][];
a[0] = new int[10];
a[1] = new int[5];
a[2] = new int[20];
```

Оператор new разрешает указывать начальные значения элементов массива с помощью *инициализатора массива*, который представляет собой список выражений, написанных между разделителями { и }.

Задание:

Создать консольное приложение – в соответствии с вариантом.

Варианты:

N Варианта	Задание
1	В данной действительной матрице размером n*m поменять местами строку, содержащую элемент с наибольшим значением, со строкой, содержащей элемент с наименьшим значением. Предполагается, что эти элементы единственны.
2	Дана действительная матрица размером n*m, все элементы которой различны. В каждой строке выбирается элемент с наибольшим значением, затем среди этих чисел выбирается наибольшее. Указать индексы элемента с найденным значением.
3	Дана целочисленная матрица размером n*m. Написать программу, формирующую двумерный массив по следующему правилу: элементы первой строки – в порядке возрастания индексов столбцов, элементы второй строки – в порядке убывания индексов столбцов и т. д.
4	Дана действительная матрица размером n*m. Найти среднее арифметическое каждого из столбцов, имеющих четные номера.
5	Дана действительная матрица размером n*m. Все элементы с наибольшим значением заменить нулями (таких элементов может быть несколько).
6	Дана целочисленная матрица размером n*m. Написать программу, позволяющую находить сумму наибольших значений элементов ее строк.
7	Дана целочисленная квадратная матрица размером n*m. Написать программу, формирующую два одномерных массива. В один переслать по строкам верхний треугольник матрицы, включая элементы главной диагонали, в другой – нижний треугольник. Полученные массивы распечатать.
8	Дана целочисленная квадратная матрица размером n*m. Написать программу, позволяющую исключать из нее столбец, в котором расположен минимальный элемент главной диагонали.
9	Дана целочисленная квадратная матрица размером n*m. Написать программу, позволяющую поменять местами

	элементы, расположенные в верхней и нижней четвертях, ограниченные главной и побочной диагоналями (за исключением элементов, расположенных на диагоналях)
10	Задана действительная матрица размером n*m. Написать программу, позволяющую заменить все элементы, наименьшие в строке, на нули.
11	Создайте двумерный массив, представляющий собой игровое поле для игры "Сапёр". Заполните его минами и определите количество мин вокруг каждой ячейки.
12	Задана целочисленная матрица размером n*m. Написать программу, позволяющую находить строки с наименьшей и наибольшей суммой и выводить их на печать.
13	Задана целочисленная квадратная матрица размером n*n. Написать программу, преобразующую исходную матрицу по правилу: начетные столбцы разделить на среднее значение диагональных элементов матрицы, а четные оставить без изменения.
14	Задана действительная квадратная матрица размером n*n. Вычислить сумму тех из ее элементов, расположенных на главной диагонали и выше ее, которые превосходят по величине все элементы, расположенные ниже главной диагонали. Если таких элементов нет, то ответом должно служить сообщение об этом.
15	Задана целочисленная квадратная матрица размером n*n (n - четное). Написать программу, позволяющую менять местами элементы первой и второй строк, элементы третьей и четвертой строк и т. д.
16	Даны две действительные квадратные матрицы размером n*n. Получить новую матрицу, прибавлением к элементам каждого столбца первой матрицы, произведения элементов соответствующих строк второй матрицы.
17	Дана целочисленная квадратная матрица размером n*n. Найти номера строк, все элементы которых – нули.
18	Задан массив из целых чисел размером n и число L. Написать программу, формирующую из него матрицу, содержащую по L элементов в строке. Недостающие элементы заполнить нулями.
19	Дана целочисленная матрица размером n*m (m - четное). Написать программу, позволяющую менять местами элементы первого и последнего столбцов, элементы второго и (n-1)-го столбцов и т. д. до среднего столбца (n - нечетно)
20	Дана действительная квадратная матрица размером n*n (n - четное), все элементы которой различны. Найти наибольший элемент среди стоящих на главной и побочной диагоналях и поменять его местами с элементом, стоящим на пересечении с этими диагоналями.

21	Дана целочисленная матрица размером n*m. Найти максимальный по модулю элемент среди отрицательных элементов нечетных столбцов.
22	Дана целочисленная матрица размером n*m и число K. Написать программу, переставляющую строки и столбцы таким образом, чтобы максимальный по модулю элемент был расположен на пересечении K–ой строки и K–го столбца.
23	Дана действительная матрица размером n*m. Все элементы с наибольшим значением заменить нулями (таких элементов может быть несколько).
24	Дана целочисленная квадратная матрица размером n*n. Написать программу, позволяющую исключать из нее столбец, в котором расположен минимальный элемент главной диагонали.
25	Дана целочисленная матрица размером n*m. Написать программу, формирующую двумерный массив по следующему правилу: элементы первой строки – в порядке возрастания индексов столбцов, элементы второй строки – в порядке убывания индексов столбцов и т. д.
26	В данной действительной матрице размером n*m обнулить все отрицательные элементы. Подсчитать, количество обнуленных элементов.
27	Задана целочисленная квадратная матрица размером n*n (n - четное). Написать программу, позволяющую менять местами элементы первой и последней строк, второй и предпоследней строк и т. д.
28	Дана целочисленная квадратная матрица размером n*n. Найти номера строк, все элементы которых отрицательны.
29	Дана действительная квадратная матрица размером n*n (n - четное), все элементы которой различны. Найти наибольший элемент среди стоящих на главной и побочной диагоналях и поменять его местами с элементом, стоящим на пересечении с этими диагоналями.
30	Даны две действительные квадратные матрицы размером n*n. Получить новую матрицу умножением элементов каждой строки первой матрицы на наибольшее из значений элементов соответствующей строки второй матрицы.

Структура отчета по лабораторной работе:

Отчет является документом, свидетельствующим о выполнении студентом лабораторной работы. Отчет должен включать:

1) Титульный лист с номером и названием лабораторной группы, идентификатором группы, номером варианта, фамилией студента, датой сдачи отчета по домашнему заданию, местом для подписи преподавателя;

2) Введение, цель и задачи лабораторной работы;
3) Описание работы программы;
4) Схему алгоритма;
5) Листинг программы (код должен быть самодокументирован, т.е. содержать комментарии);
6) Демонстрацию работы приложения (скриншоты ручного тестирования)
7) Вывод

Структура отчета по лабораторной работе:

Отчет является документом, свидетельствующим о выполнении студентом лабораторной работы. Отчет должен включать:
1) Титульный лист с номером и названием лабораторной группы, идентификатором группы, номером варианта, фамилией студента, датой сдачи отчета по домашнему заданию, местом для подписи преподавателя;
2) Введение, цель и задачи лабораторной работы;
3) Описание работы программы;
4) Схему алгоритма;
5) Листинг программы (код должен быть самодокументирован, т.е. содержать комментарии);
6) Демонстрацию работы приложения (скриншоты ручного тестирования)
7) Вывод

Контрольные вопросы:

1) Чем отличается одномерный массив от двумерного массива в C#? Приведите примеры использования каждого из них.
2) Как объявить и инициализировать одномерный массив в C#? Приведите пример и объясните, как получить доступ к элементам массива.
3) Что такое индексация в контексте массивов? Какие значения могут быть использованы в качестве индексов для доступа к элементам массива?
4) Как создать и заполнить двумерный массив в C#? Предоставьте пример объявления и инициализации двумерного массива, а также доступа к его элементам.
5) Какие методы и свойства предоставляет класс Array в C# для работы с массивами? Укажите хотя бы два метода и два свойства и объясните их использование.

Лабораторная работа №3. Классы

Теоретическая часть:

Классы. Классы в C# имеют тот же смысл, что и в языках C++ и Java.

Формально класс описывается следующим образом:

```
модификатор-доступа class Имя-класса {
    … описание данных и методов …
}
```

Следующий код является простым примером объявления класса с именем Point:

```
public class Point
{
    public int X { get; }
    public int Y { get; }

    public Point(int x, int y) => (X, Y) = (x, y);
}
```

Экземпляры классов создаются с помощью оператора new, который выделяет память для нового экземпляра, вызывает конструктор для инициализации этого экземпляра и возвращает ссылку на экземпляр.

Некоторые методы и свойства специально предназначены для того, чтобы их вызов или доступ к ним осуществлялся из *клиентского кода*, то есть из кода за пределами этого класса или структуры. Другие методы и свойства могут использоваться только в самом классе или структуре.

Важно ограничить доступность кода так, чтобы только нужные элементы клиентского кода получали к нему доступ. Уровень доступности для типов и их элементов вы можете задать с помощью следующих модификаторов доступа.

- public
- protected
- internal
- protected internal
- private
- private protected.

По умолчанию используется режим доступа private.

Задание:

Создать консольное приложение – в соответствии с вариантом.

Варианты:

N Варианта	Задание
1	Создать консольное приложение для учета банковских сведений клиентов. Программа представляет собой автоматизированную систему учета банковских сведений. На каждого клиента банка хранятся следующие сведения: Ф.И.О., возраст, место работы, номера счетов. На каждом счете хранится информация о текущем балансе и история прихода и расхода. Для каждого клиента может быть создано неограниченное количество счетов. С каждым счетом можно производить следующие действия: открытие, закрытие, вклад денег, снятие денег, просмотр баланса, просмотр истории. Вся информация должна храниться в массивах. Рекомендуется объекты клиента и счета реализовать в виде классов. Баланс счета организовать в виде свойства только для чтения.
2	Создать консольное приложение для учета личных контактов. Программа позволяет хранить информацию о контактах, включая Ф.И.О., возраст, место работы и контактные данные. Для каждого контакта можно хранить дополнительную информацию, такую как дни рождения и заметки. Реализовать возможность добавления новых контактов, редактирования существующих, удаления и поиска по различным параметрам..
3	Разработать консольное приложение для учета недвижимости. Программа позволяет хранить информацию о недвижимых объектах, включая адрес, тип (квартира, дом, земля), стоимость и текущего владельца. Для каждого объекта можно регистрировать изменения владельцев и проводить анализ рынка недвижимости. Реализовать функциональность добавления новых объектов, изменения владельца, вычисления общей стоимости объектов и поиска объектов по адресу.
4	Создать консольное приложение для учета автомобилей. Программа позволяет хранить информацию о каждом автомобиле, включая марку, модель, год выпуска, цвет и номер. Для каждого автомобиля можно регистрировать технические осмотры, проводить расчеты затрат на топливо и техническое обслуживание. Реализовать возможность добавления новых автомобилей, внесения данных о расходе топлива, а также анализа расходов на техническое обслуживание.
5	Разработать консольное приложение для учета задач и проектов

	в офисе. Программа позволяет хранить информацию о задачах, включая название, описание, статус выполнения, ответственного сотрудника и срок выполнения. Для каждой задачи можно фиксировать изменения статуса и комментарии. Реализовать функциональность добавления новых задач, назначения ответственного, изменения статуса и отображения задач по ответственному и сортировки по сроку выполнения.
6	Разработать консольное приложение для учета задач и проектов в медицинской клинике. Программа позволяет хранить информацию о задачах, связанных с пациентами и медицинскими процедурами, включая название задачи, описание, статус выполнения, ответственного медицинского работника и срок выполнения. Реализовать возможность добавления новых задач, назначения ответственных, изменения статусов и отображения задач по медицинскому работнику.
7	Создать консольное приложение для учета задач и проектов в отделе рекламы и маркетинга. Программа позволяет хранить информацию о задачах, связанных с рекламными кампаниями и маркетинговыми мероприятиями, включая название задачи, описание, статус выполнения, ответственного сотрудника и срок выполнения. Реализовать функциональность добавления новых задач, назначения ответственных, изменения статусов и отображения задач по сотруднику.
8	Разработать консольное приложение для учета задач и проектов в сфере информационной безопасности. Программа позволяет хранить информацию о задачах, связанных с анализом угроз, мониторингом безопасности и реагированием на инциденты, включая название задачи, описание, статус выполнения, ответственного аналитика и срок выполнения. Реализовать возможность добавления новых задач, назначения ответственных, изменения статусов и отображения задач по аналитику.
9	Создать консольное приложение для учета задач и проектов в отделе кадров. Программа позволяет хранить информацию о задачах, связанных с наймом, увольнением, обучением сотрудников и управлением персоналом, включая название задачи, описание, статус выполнения, ответственного HR-специалиста и срок выполнения. Реализовать функциональность добавления новых задач, назначения ответственных, изменения статусов и отображения задач по HR-специалисту.
10	Разработать консольное приложение для учета задач и проектов в области исследований и разработок. Программа позволяет хранить информацию о задачах, связанных с исследовательскими проектами, включая название задачи,

	описание, статус выполнения, ответственного исследователя и срок выполнения. Реализовать возможность добавления новых задач, назначения ответственных, изменения статусов и отображения задач по исследователю.
11	Создать консольное приложение для учета задач и проектов в отделе логистики. Программа позволяет хранить информацию о задачах, связанных с управлением поставками, складским хранением и распределением товаров, включая название задачи, описание, статус выполнения, ответственного логиста и срок выполнения. Реализовать функциональность добавления новых задач, назначения ответственных, изменения статусов и отображения задач по логисту.
12	Разработать консольное приложение для учета задач и проектов в отделе обслуживания клиентов. Программа позволяет хранить информацию о задачах, связанных с обработкой запросов и жалоб клиентов, включая название задачи, описание, статус выполнения, ответственного сотрудника по обслуживанию клиентов и срок выполнения. Реализовать возможность добавления новых задач, назначения ответственных, изменения статусов и отображения задач по сотруднику по обслуживанию клиентов.
13	Создать консольное приложение для учета задач и проектов в отделе продаж. Программа позволяет хранить информацию о задачах, связанных с продажами, включая название задачи, описание, статус выполнения, ответственного менеджера по продажам и срок выполнения. Реализовать функциональность добавления новых задач, назначения ответственных, изменения статусов и отображения задач по менеджеру по продажам.
14	Разработать консольное приложение для учета задач и проектов в отделе закупок. Программа позволяет хранить информацию о задачах, связанных с закупками товаров и материалов, включая название задачи, описание, статус выполнения, ответственного специалиста по закупкам и срок выполнения. Реализовать возможность добавления новых задач, назначения ответственных, изменения статусов и отображения задач по специалисту по закупкам.
15	Создать консольное приложение для учета задач и проектов в отделе качества. Программа позволяет хранить информацию о задачах, связанных с контролем качества продукции или услуг, включая название задачи, описание, статус выполнения, ответственного специалиста по контролю качества и срок выполнения. Реализовать функциональность добавления новых задач, назначения ответственных, изменения статусов и отображения задач по специалисту по контролю качества.
16	Разработать консольное приложение для учета задач и проектов

	в отделе технической поддержки. Программа позволяет хранить информацию о задачах, связанных с обработкой запросов и устранением неисправностей, включая название задачи, описание, статус выполнения, ответственного специалиста по технической поддержке и срок выполнения. Реализовать возможность добавления новых задач, назначения ответственных, изменения статусов и отображения задач по специалисту по технической поддержке.
17	Создать консольное приложение для учета задач и проектов в отделе образования. Программа позволяет хранить информацию о задачах, связанных с образовательными мероприятиями, включая название задачи, описание, статус выполнения, ответственного преподавателя и срок выполнения. Реализовать функциональность добавления новых задач, назначения ответственных, изменения статусов и отображения задач по преподавателю.
18	Разработать консольное приложение для учета задач и проектов в отделе исследований и разработок. Программа позволяет хранить информацию о задачах, связанных с научными исследованиями и разработкой новых продуктов, включая название задачи, описание, статус выполнения, ответственного исследователя и срок выполнения. Реализовать возможность добавления новых задач, назначения ответственных, изменения статусов и отображения задач по исследователю.
19	Создать консольное приложение для учета задач и проектов в отделе креативных искусств. Программа позволяет хранить информацию о задачах, связанных с созданием искусственных произведений, включая название задачи, описание, статус выполнения, ответственного художника и срок выполнения. Реализовать функциональность добавления новых задач, назначения ответственных, изменения статусов и отображения задач по художнику.
20	Разработать консольное приложение для учета задач и проектов в отделе архитектуры и дизайна. Программа позволяет хранить информацию о задачах, связанных с созданием архитектурных проектов и дизайнерскими работами, включая название задачи, описание, статус выполнения, ответственного архитектора или дизайнера и срок выполнения. Реализовать возможность добавления новых задач, назначения ответственных, изменения статусов и отображения задач по архитектору или дизайнеру
21	Создать консольное приложение для учета задач и проектов в отделе строительства. Программа позволяет хранить информацию о задачах, связанных с строительными проектами, включая название задачи, описание, статус выполнения, ответственного инженера или строительного менеджера и срок

	выполнения. Реализовать функциональность добавления новых задач, назначения ответственных, изменения статусов и отображения задач по инженеру или строительному менеджеру.
22	Разработать консольное приложение для учета задач и проектов в отделе медицинских исследований. Программа позволяет хранить информацию о задачах, связанных с клиническими исследованиями и медицинскими проектами, включая название задачи, описание, статус выполнения, ответственного медицинского исследователя и срок выполнения. Реализовать возможность добавления новых задач, назначения ответственных, изменения статусов и отображения задач по медицинскому исследователю.
23	Создать консольное приложение для учета задач и проектов в отделе тестирования программного обеспечения. Программа позволяет хранить информацию о задачах, связанных с тестированием приложений и выявлением ошибок, включая название задачи, описание, статус выполнения, ответственного тестировщика и срок выполнения. Реализовать функциональность добавления новых задач, назначения ответственных, изменения статусов и отображения задач по тестировщику.
24	Разработать консольное приложение для учета задач и проектов в отделе экологии и охраны окружающей среды. Программа позволяет хранить информацию о задачах, связанных с мониторингом экологических параметров и выполнением проектов по охране окружающей среды, включая название задачи, описание, статус выполнения, ответственного эколога и срок выполнения. Реализовать возможность добавления новых задач, назначения ответственных, изменения статусов и отображения задач по экологу.
25	Создать консольное приложение для учета задач и проектов в отделе правовых вопросов. Программа позволяет хранить информацию о задачах, связанных с юридическими консультациями, анализом законодательства и юридическим сопровождением проектов, включая название задачи, описание, статус выполнения, ответственного юриста и срок выполнения. Реализовать функциональность добавления новых задач, назначения ответственных, изменения статусов и отображения задач по юристу.
26	Разработать консольное приложение для учета задач и проектов в отделе исследования рынка. Программа позволяет хранить информацию о задачах, связанных с анализом рыночных тенденций, конкурентной среды и потребительских предпочтений, включая название задачи, описание, статус выполнения, ответственного аналитика по исследованию рынка

	и срок выполнения. Реализовать возможность добавления новых задач, назначения ответственных, изменения статусов и отображения задач по аналитику.
27	Создать консольное приложение для учета задач и проектов в отделе обслуживания оборудования. Программа позволяет хранить информацию о задачах, связанных с техническим обслуживанием и ремонтом оборудования, включая название задачи, описание, статус выполнения, ответственного техника и срок выполнения. Реализовать функциональность добавления новых задач, назначения ответственных, изменения статусов и отображения задач по технику.
28	Разработать консольное приложение для учета задач и проектов в отделе безопасности. Программа позволяет хранить информацию о задачах, связанных с обеспечением безопасности и мониторингом угроз, включая название задачи, описание, статус выполнения, ответственного специалиста по безопасности и срок выполнения. Реализовать возможность добавления новых задач, назначения ответственных, изменения статусов и отображения задач по специалисту по безопасности.
29	Создать консольное приложение для учета задач и проектов в отделе туризма и гостеприимства. Программа позволяет хранить информацию о задачах, связанных с организацией туристических мероприятий, бронированием отелей и обслуживанием гостей, включая название задачи, описание, статус выполнения, ответственного менеджера по туризму и срок выполнения. Реализовать функциональность добавления новых задач, назначения ответственных, изменения статусов и отображения задач по менеджеру по туризму.
30	Разработать консольное приложение для учета задач и проектов в отделе информационных технологий. Программа позволяет хранить информацию о задачах, связанных с разработкой, тестированием и обслуживанием программного обеспечения, включая название задачи, описание, статус выполнения, ответственного IT-специалиста и срок выполнения. Реализовать возможность добавления новых задач, назначения ответственных, изменения статусов и отображения задач по IT-специалисту.

Структура отчета по лабораторной работе:

Отчет является документом, свидетельствующим о выполнении студентом лабораторной работы. Отчет должен включать:

1) Титульный лист с номером и названием лабораторной группы, идентификатором группы, номером варианта, фамилией студента, датой сдачи отчета по домашнему заданию, местом для подписи преподавателя;

2) Введение, цель и задачи лабораторной работы;
3) Описание работы программы;
4) Схему алгоритма;
5) Листинг программы (код должен быть самодокументирован, т.е. содержать комментарии);
6) Демонстрацию работы приложения (скриншоты ручного тестирования)
7) Вывод

Структура отчета по лабораторной работе:

Отчет является документом, свидетельствующим о выполнении студентом лабораторной работы. Отчет должен включать:
1) Титульный лист с номером и названием лабораторной группы, идентификатором группы, номером варианта, фамилией студента, датой сдачи отчета по домашнему заданию, местом для подписи преподавателя;
2) Введение, цель и задачи лабораторной работы;
3) Описание работы программы;
4) Схему алгоритма;
5) Листинг программы (код должен быть самодокументирован, т.е. содержать комментарии);
6) Демонстрацию работы приложения (скриншоты ручного тестирования)
7) Вывод

Контрольные вопросы:

1) Что такое класс в C# и какова его роль в объектно-ориентированном программировании?
2) Как объявить класс в C#? Приведите пример объявления простого класса.

3) Что такое конструктор класса? Какие типы конструкторов существуют в C#?
4) Какие модификаторы доступа могут быть использованы для членов класса, и как они влияют на доступ к этим членам?
5) Какие ключевые слова используются для наследования в C#? Как создать производный класс (наследник) от другого класса?
6) Что такое поля (переменные-члены) класса и методы класса? Как они отличаются друг от друга?
7) Как можно обратиться к членам класса извне класса и внутри класса? Какие ключевые слова используются для этого?
8) Что такое статические члены класса, и как они отличаются от экземплярных членов?
9) Какие методы и свойства класса могут быть объявлены как абстрактные, и какие классы могут содержать абстрактные члены?
10) Что такое инкапсуляция, наследование и полиморфизм, и как они связаны с классами в объектно-ориентированном программировании?

Лабораторная работа №4. Windows Forms

Теоретическая часть:

Windows Forms — это платформа пользовательского интерфейса для создания классических приложений Windows. Она обеспечивает один из самых эффективных способов создания классических приложений с помощью визуального конструктора в Visual Studio. Такие функции, как размещение визуальных элементов управления путем перетаскивания, упрощают создание классических приложений.

В Windows Forms можно разрабатывать графически сложные приложения, которые просто развертывать, обновлять, и с которыми удобно работать как в автономном режиме, так и в сети. Приложения Windows Forms могут получать доступ к локальному оборудованию и файловой системе компьютера, на котором работает приложение.

На приведенном ниже рисунке показан шаблон проекта Windows Forms с C#:

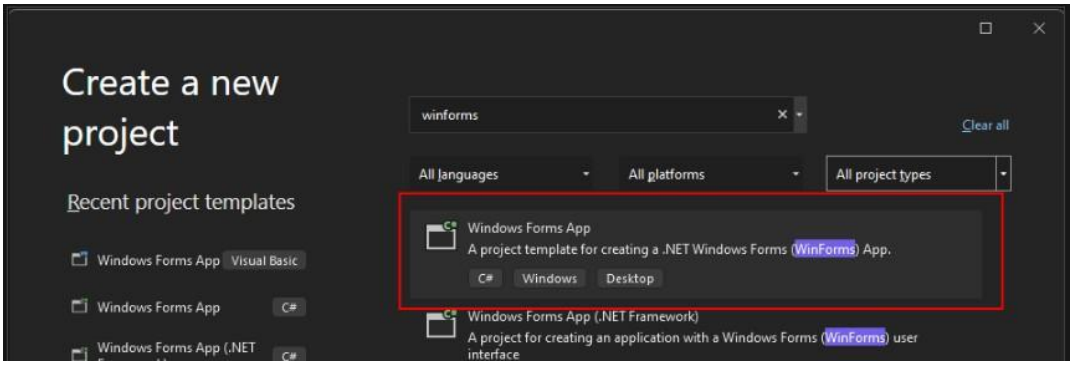

Рисунок 2 – Шаблон проекта

Поддержка Windows Forms в Visual Studio состоит из четырех важных компонентов, с которыми вы будете взаимодействовать при создании приложения.

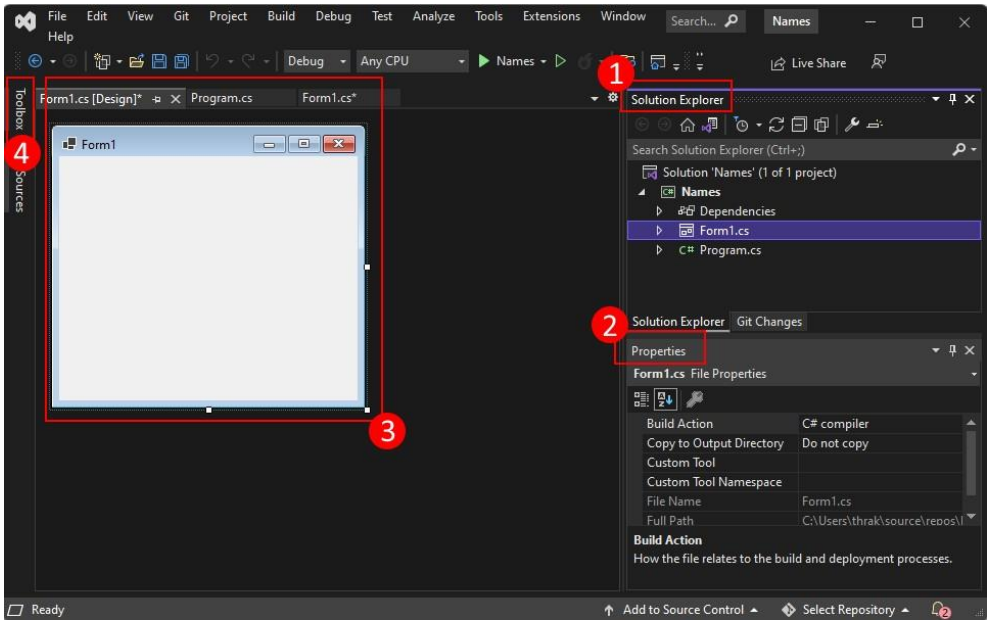

Рисунок 3 – Компоненты Windows Forms

1. Обозреватель решений

   Все файлы проекта, код, формы и ресурсы отображаются в этой области.
2. Properties (Свойства)

На этой панели отображаются параметры свойств, которые можно настроить в зависимости от выбранного элемента. Например, если выбрать элемент в **Обозревателе решений**, отобразятся параметры свойств, связанные с файлом. Если выбрать объект в **конструкторе**, отобразятся параметры элемента управления или формы.

3. Конструктор форм

Это конструктор для формы. Он является интерактивным, и на него можно перетаскивать объекты из **панели элементов**. Выбирая и перемещая элементы в конструкторе, можно визуально создавать пользовательский интерфейс для приложения.

4. Панель элементов

Панель элементов содержит все элементы управления, которые можно добавить на форму. Чтобы добавить элемент управления на текущую форму, дважды щелкните элемент управления или перетащите его.

Задание:

Создать интерфейс к программе, созданной в лабораторной работе 3. Ввод сведений реализовать через дополнительные формы.

Структура отчета по лабораторной работе:

Отчет является документом, свидетельствующим о выполнении студентом лабораторной работы. Отчет должен включать:

1) Титульный лист с номером и названием лабораторной группы, идентификатором группы, номером варианта, фамилией студента, датой сдачи отчета по домашнему заданию, местом для подписи преподавателя;
2) Введение, цель и задачи лабораторной работы;
3) Описание работы программы;
4) Схему алгоритма;
5) Листинг программы (код должен быть самодокументирован, т.е. содержать комментарии);
6) Демонстрацию работы приложения (скриншоты ручного тестирования)
7) Вывод

Контрольные вопросы:

1) Что такое Windows Forms (WinForms) в C# и для чего они используются?
2) Как создать новое Windows Forms приложение в Visual Studio?
3) Какие элементы управления (контролы) доступны в Windows Forms, и как они различаются по функциональности?
4) Как можно связать события (event) с элементами управления в Windows Forms, и как обработать событие клика на кнопке?

5) Как осуществить навигацию между формами (окнами) в Windows Forms приложении?
6) Как передавать данные между формами в Windows Forms приложении?
7) Как создать пользовательский интерфейс (UI) с использованием Windows Forms, включая размещение элементов управления, изменение их свойств и стилей?
8) Как добавить меню и обработать выбор пунктов меню в Windows Forms приложении?
9) Как реализовать валидацию ввода данных в текстовых полях (TextBox) в Windows Forms?
10) Как сохранить и загрузить данные из файлов в Windows Forms приложении с использованием диалоговых окон для выбора файлов?.

Лабораторная работа №5. Файлы

Задание:

Создать консольное приложение – в соответствии с вариантом.

Варианты:

N Варианта	Задание
1	Дан символьный файл F, в котором не менее двух компонентов. Определить, являются ли два первых символа файла цифрами. Если да, то установить, является ли число, образованное этими цифрами, четным.
2	Даны символьные файлы F и G. Определить, совпадают ли компоненты файла F с компонентами файла G. Если нет, то получить номер первого несовпадающего компонента.
3	Дан текстовый файл F. Найти и вывести на экран все строки, в которых присутствует заданное ключевое слово.
4	Создать программу, которая открывает текстовый файл F, подсчитывает количество слов в нем и выводит результат на экран.
5	Написать функцию, которая создает новый текстовый файл, содержащий только уникальные строки из исходного файла F.
6	Дан текстовый файл F с числами. Найти среднее арифметическое всех чисел в файле и вывести его на экран.
7	Дан текстовый файл F, в котором записаны имена студентов и их оценки. Создать новый файл, в котором будут только имена студентов, получивших оценку "5".
8	Напишите функцию, которая определяет, сколько раз заданное слово встречается в текстовом файле F.
9	Дан текстовый файл F с числами. Найти минимальное и максимальное число в файле и вывести их на экран.
10	Создать программу, которая копирует содержимое одного текстового файла F в другой текстовый файл G.
11	Дан текстовый файл F, содержащий даты в формате "день.месяц.год". Определить, сколько из них соответствуют заданному месяцу и вывести на экран.
12	Написать функцию, которая считает сумму всех чисел в текстовом файле F и выводит результат на экран.
13	Дан текстовый файл F, в котором записаны имена и возрасты людей. Создать новый файл, в котором будут только имена людей старше 18 лет.
14	Разработать программу, которая считывает текстовый файл F и подсчитывает количество символов в каждой строке, затем выводит результат на экран.

15	Дан текстовый файл F с данными о продажах товаров. Создать новый файл, в котором будут только записи о продажах, превышающие заданную сумму.
16	Напишите функцию, которая определяет, сколько раз каждое слово встречается в текстовом файле F, и выводит результат на экран.
17	Дан текстовый файл F с информацией о книгах (название, автор, год издания). Создать новый файл, в котором будут только записи книг определенного автора.
18	Создать программу, которая читает текстовый файл F, удаляет из него все пустые строки и сохраняет результат в новый файл.
19	Дан текстовый файл F с данными о сотрудниках (фамилия, имя, возраст). Создать новый файл, в котором будут только записи сотрудников, младше 30 лет.
20	Написать функцию, которая определяет, сколько раз в текстовом файле F встречается определенный символ, и выводит результат на экран.
21	Дан текстовый файл F с информацией о фильмах (название, режиссер, год выпуска). Создать новый файл, в котором будут только записи фильмов определенного режиссера.
22	Создать программу, которая считывает текстовый файл F и выводит на экран самое длинное слово в файле.
23	Дан текстовый файл F с данными о заказах (номер заказа, сумма). Создать новый файл, в котором будут только заказы с суммой больше определенной.
24	Написать функцию, которая определяет, сколько раз каждая буква встречается в текстовом файле F, и выводит результат на экран.
25	Дан текстовый файл F с информацией о клиентах (фамилия, адрес, телефон). Создать новый файл, в котором будут только записи клиентов из определенного города.
26	Создать программу, которая считывает текстовый файл F и выводит на экран количество слов, начинающихся с определенной буквы.
27	Дан текстовый файл F с данными о продуктах (название, цена). Создать новый файл, в котором будут только записи о продуктах с ценой ниже заданной.
28	Напишите функцию, которая определяет, сколько раз в текстовом файле F встречается заданное слово, и выводит результат на экран.
29	Дан текстовый файл F с информацией о городах (название, население). Создать новый файл, в котором будут только записи о городах с населением выше определенного уровня.
30	Дан текстовый файл F. Найти и вывести на экран все строки, в

	которых присутствует заданное ключевое слово.

Структура отчета по лабораторной работе:

Отчет является документом, свидетельствующим о выполнении студентом лабораторной работы. Отчет должен включать:

1) Титульный лист с номером и названием лабораторной группы, идентификатором группы, номером варианта, фамилией студента, датой сдачи отчета по домашнему заданию, местом для подписи преподавателя;
2) Введение, цель и задачи лабораторной работы;
3) Описание работы программы;
4) Схему алгоритма;
5) Листинг программы (код должен быть самодокументирован, т.е. содержать комментарии);
6) Демонстрацию работы приложения (скриншоты ручного тестирования)
7) Вывод

Контрольные вопросы:

1) Что такое тип делегата? Какой аналог типа делегата существует в C#?
2) Опишите основные направления использования делегатов.
3) Какие механизмы технологии Windows Forms реализованы с использованием делегатов?
4) Для чего предназначен тип Action<T>? Чем он отличается от Func<T>?
5) Чем пользовательские делегаты отличаются от библиотечных?

Лабораторная работа №6. Делегаты

Теоретическая часть:

Делегат — это тип, который представляет ссылки на методы с определенным списком параметров и типом возвращаемого значения. При создании экземпляра делегата этот экземпляр можно связать с любым методом с совместимой сигнатурой и типом возвращаемого значения. Метод можно вызвать (активировать) с помощью экземпляра делегата.

Делегаты используются для передачи методов в качестве аргументов к другим методам. Обработчики событий — это ничто иное, как методы, вызываемые с помощью делегатов. При создании пользовательского метода класс (например, элемент управления Windows) может вызывать этот метод при появлении определенного события. В следующем примере показано объявление делегата:

```
public delegate int PerformCalculation(int x, int y);
```

Делегату можно назначить любой метод из любого доступного класса или структуры, соответствующей типу делегата. Этот метод должен быть статическим методом или методом экземпляра. Такая гибкость позволяет программно изменять вызовы метода, а также включать новый код в существующие классы.

Делегаты имеют следующие свойства.

- Делегаты подобны указателям на функции в C++, но являются полностью объектно-ориентированными и, в отличие от указателей C++ на функции-члены, инкапсулируют экземпляр объекта вместе с методом.
- Делегаты допускают передачу методов в качестве параметров.
- Делегаты можно использовать для определения методов обратного вызова.
- Делегаты можно связывать друг с другом; например, при появлении одного события можно вызывать несколько методов.
- Точное соответствие методов типу делегата не требуется. Для краткой записи встроенных блоков кода введены лямбда-выражения. В результате компиляции лямбда-выражений (в определенном контексте) получаются типы делегатов.

Задание:

Создать консольное приложение в соответствии с индивидуальным вариантом разработайте требуемый тип делегата (пользовательский, библиотечный или лямбда-выражение).

Варианты:

N	Тип делегата	Задание	Входные параметры
1	пользовательский	Метод возвращает сумму элементов матрицы целых случайных чисел	Два параметра: размер матрицы
2	библиотечный	Метод возвращает разницу максимального и минимального элементов матрицы целых случайных чисел	Два параметра: размер матрицы
3	лямбда-выражение	Метод возвращает логическое значение, указывающее существует ли заданное число в массиве целых случайных чисел	Два параметра: размер массива и искомый элемент

4	пользовательский	Метод возвращает матрицу случайных битовых значений (0 или 1), формируемую случайным образом	Два параметра: размер матрицы
5	библиотечный	Метод возвращает строку максимальной длины на основе переданного массива строк	Один параметр: массив (или список) строк
6	лямбда-выражение	Метод возвращает подмножество элементов массива случайных чисел, которые делятся на 3	Один параметр: размер исходного массива
7	пользовательский	Метод возвращает число – количество элементов присутствующих в обоих массивах случайных чисел	Два параметра: размеры массивов
8	библиотечный	Метод возвращает среднее арифметическое элементов матрицы случайных чисел.	Два параметра: размер матрицы
9	лямбда-выражение	Метод возвращает результат шифрования строки: каждый исходный символ строки заменяется шифрованным символом, код которого	Два параметра: исходная строка, число сдвига n
10	пользовательский	Метод возвращает статистику для строки: список пар (символ строки, количество вхождений)	Один параметр: анализируемая строка
11	библиотечный	Метод возвращает разницу максимального и минимального элементов матрицы.	Два параметра: размер матрицы
12	лямбда-выражение	Метод возвращает подмножество элементов массива случайных чисел, которые являются четными	Один параметр: размер исходного массива
13	пользовательский	Метод возвращает количество нечетных элементов в матрице случайных чисел	Два параметра: размер

			матрицы
14	библиотечный	Метод возвращает подмножество элементов массива случайных чисел, которые отличаются от заданного числа не более чем на 4.	Два параметра: размер массива, значеное число
15	лямбда-выражение	Метод возвращает логическое значение, указывающее существует ли заданный символ в строке	Два параметра: строка и искомый символ
16	пользовательский	Метод возвращает два числа – максимальные элементы массивов случайных чисел.	Два параметра: размеры массивов
17	библиотечный	Метод возвращает скадярное произведение двух случайных векторов	Один параметр: размер векторов
18	лямбда-выражение	Метод возвращает сумму двух матриц случайных чисел	Два параметра: размер матриц
19	пользовательский	Метод возвращает количество вхождений заданного элемента в матрице вещественных чисел	Два параметра: матрица и целевой элемент
20	библиотечный	Метод возвращает подмножество элементов массива случайных чисел, которые делятся на 6	Один параметр: размер исходного массива
21	лямбда-выражение	Метод возвращает результат сложения двух случайных векторов целых чисел	Один параметр: размер векторов
22	пользовательский	Метод возвращает количество вхождений заданного символа в строке	Два параметра: строка и целевой символ
23	библиоте	Метод возвращает подмножество	Два

	чный	четных элементов матрицы случайных чисел	параметра: размер матрицы
24	лямбда-выражение	Метод возвращает статистику массива случайных целых чисел: список пар (число массива, количество вхождений)	Один параметр: размер массива
25	пользовательский	Метод возвращает логическое значение, указывающее существует ли заданный элемент в матрице чисел double	Два параметра: матрица и искомый элемент

Структура отчета по лабораторной работе:

Отчет является документом, свидетельствующим о выполнении студентом лабораторной работы. Отчет должен включать:

1) Титульный лист с номером и названием лабораторной группы, идентификатором группы, номером варианта, фамилией студента, датой сдачи отчета по домашнему заданию, местом для подписи преподавателя;
2) Введение, цель и задачи лабораторной работы;
3) Описание работы программы;
4) Схему алгоритма;
5) Листинг программы (код должен быть самодокументирован, т.е. содержать комментарии);
6) Демонстрацию работы приложения (скриншоты ручного тестирования)
7) Вывод

Контрольные вопросы:

1) Что такое делегат в C# и для чего он используется?
2) Как объявить и инициализировать делегат в C#?
3) Какие основные типы делегатов предоставляются в C#?
4) Какие основные различия между делегатами и интерфейсами?
5) Как можно использовать анонимные методы с делегатами в C#?

Лабораторная работа №7. Рефлексия

Теоретическая часть:

Механизм отражения позволяет получать объекты (типа Type), которые описывают сборки, модули и типы. Отражение можно использовать для динамического создания экземпляра типа, привязки типа к существующему объекту, а также получения типа из существующего объекта и вызова его методов или доступа к его полям и свойствам. Если в коде используются атрибуты, отражение обеспечивает доступ к ним.

Вот простой пример отражения, в котором для получения типа переменной используется метод GetType(), наследуемый всеми типами от базового класса Object:

```
// Using GetType to obtain type information:
int i = 42;
Type type = i.GetType();
Console.WriteLine(type);
```

Выходные данные: *System.Int32*.

В этом примере отражение используется для получения полного имени загруженной сборки.

```
// Using Reflection to get information of an Assembly:
Assembly info = typeof(int).Assembly;
Console.WriteLine(info);
```

Выходные данные: *System.Private.CoreLib, Version=4.0.0.0, Culture=neutral, PublicKeyToken=7cec85d7bea7798e.*

Отражение удобно использовать в следующих ситуациях:

- При необходимости доступа к атрибутам в метаданных программы.
- Для проверки и создания экземпляров типов в сборке.
- Для создания типов во время выполнения.
- Для выполнения позднего связывания, которое обеспечивает доступ к методам в типах, созданных во время выполнения.

Задание:

Разработать программу, реализующую работу с рефлексией.

1. Программа должна быть разработана в виде консольного приложения на языке C#.

2. Создайте класс, содержащий конструкторы, свойства, методы.

3. С использованием рефлексии выведите информацию о конструкторах, свойствах, методах.

4. Создайте класс атрибута (унаследован от класса System.Attribute).

5. Назначьте атрибут некоторым свойствам классам. Выведите только те свойства, которым назначен атрибут.

6. Вызовите один из методов класса с использованием рефлексии

Лабораторная работа №8. Исключения

Теоретическая часть:

Функции обработки исключений в языке C# помогают вам справиться с непредвиденными или исключительными проблемами, которые возникают при выполнении программы. При обработке исключений используются ключевые слова try, catch и finally для действий, которые могут оказаться неудачными. Это позволяет обрабатывать ошибки так, как кажется разумным, а также правильно высвобождать ресурсы. Исключения могут создаваться средой выполнения (CLR), платформой .NET , библиотеками сторонних поставщиков или кодом самого приложения. Чтобы создать исключение, используйте ключевое слово throw.

Во многих случаях исключение может создаваться не тем методом, который вызывается в вашем коде, а одним из последующих методов в стеке вызовов. Если создается такое исключение, среда CLR разворачивает стек, находит метод с блоком catch для исключений соответствующего типа и выполняет первый такой обнаруженный блок catch. Если подходящий блок catch не будет обнаружен во всем стеке вызовов, среда CLR завершает процесс и выводит сообщение для пользователя.

В этом примере метод выполняет проверку деления на нуль и перехватывает ошибку. Если не использовать обработку исключений, такая программа завершит работу с ошибкой **DivideByZeroException was unhandled** (Исключение DivideByZero не обработано).

```
public class ExceptionTest
{
    static double SafeDivision(double x, double y)
    {
        if (y == 0)
            throw new DivideByZeroException();
        return x / y;
    }

    public static void Main()
    {
        // Input for test purposes. Change the values to see
        // exception handling behavior.
        double a = 98, b = 0;
        double result;

        try
        {
            result = SafeDivision(a, b);
            Console.WriteLine("{0} divided by {1} = {2}", a, b, result);
        }
        catch (DivideByZeroException)
        {
            Console.WriteLine("Attempted divide by zero.");
        }
    }
}
```

Исключения имеют следующие свойства.

- Исключения представляют собой типы, производные в конечном счете от System.Exception.
- Используйте блок try для выполнения таких инструкций, которые могут создавать исключения.
- Когда внутри такого блока try возникает исключение, поток управления переходит к первому подходящему обработчику исключений в стеке вызовов. В C# ключевое слово catch обозначает обработчик исключений.
- Если для созданного исключения не существует обработчиков, выполнение программы прекращается с сообщением об ошибке.
- Не перехватывайте исключение, если вы не намерены его обрабатывать с сохранением известного состояния приложения. Если вы перехватываете System.Exception, создайте его заново в конце блока catch, используя ключевое слово throw.
- Если блок catch определяет переменную исключения, ее можно использовать для получения дополнительных сведений о типе созданного исключения.
- Программа может явным образом создавать исключения с помощью ключевого слова throw.
- Объекты исключения содержат подробные сведения об ошибке, например состояние стека вызовов и текстовое описание ошибки.
- Код в блоке finally выполняется, даже если создано исключение. Используйте блок finally, чтобы высвободить ресурсы, например закрыть потоки и файлы, которые были открыты внутри блока try.
- Управляемые исключения реализованы в платформе .NET на основе структурированного механизма обработки исключений Win32.

Задание:

Составить программу деления вещественных чисел. программа должна выполнять обработку исключений с использованием конструкции try … catch, и выдавать следующие сообщения о характере ошибки:

1. не введено число (с помощью оператора условия);
2. введено слишком длинное число (с помощью оператора условия);
3. деление на ноль;
4. ошибка преобразования.

Лабораторная работа №9. Сериализация

Теоретическая часть:

Сериализация представляет собой процесс преобразования состояния объекта в форму, пригодную для сохранения или передачи. Дополнением к сериализации служит десериализация, при которой осуществляется преобразование потока в объект. Вместе эти процессы обеспечивают хранение и передачу данных.

В .NET доступны следующие технологии сериализации:

- При двоичной сериализации сохраняется правильность типов, что полезно для сохранения состояния объекта между разными вызовами приложения. Например, можно обеспечить совместный доступ к объекту для разных приложений, сериализовав его в буфер обмена. Объект можно сериализовать в поток, на диск, в память, передать по сети и т. д. При удаленном управлении сериализация используется для передачи объектов "по значению" с одного компьютера или домена приложения на другой.
- При сериализации XML и SOAP сериализуются только открытые свойства и поля, а правильность типов не сохраняется. Этот метод полезен для предоставления или использования данных без ограничений работающего с ними приложения. Будучи открытым стандартом, XML привлекателен для совместного использования данных в Интернете. Аналогичным образом и SOAP представляет собой открытый стандарт, использование которого эффективно и удобно.
- При сериализации JSON сериализуются только открытые свойства, а правильность типов не сохраняется. Будучи открытым стандартом, JSON привлекателен для совместного использования данных в Интернете.

Задание:

Требуется разработать программу, ведущую учёт заказов в магазине. Классы:

- (Покупатель) с тремя атрибутами: имя (string), адрес (string), скидка (double)
- (Товар). Поля, соответствующие названию (string) и цене (decimal)
- (База данных товаров), хранящий ассоциативный массив («словарь») с информацией о товарах
- OrderLine с полями количество (int) и продукт (Product)
- Order с полями номер заказа (int), клиент (Customer), скидка (decimal), общая стоимость (decimal) и строки заказа (List<OrderLIne>).

Реализовать следующую логику основной программы:

1. Создаётся и заполняется база данных товаров (ассоциативный массив).
2. В консоли вводятся данные по конкретному покупателю, создаётся соответствующий объект.
3. Создаётся заказ для введённого ранее покупателя. Устанавливается скидка на заказ в соответствии со скидкой покупателя.
4. В цикле формируются необходимое количество строк заказа: вводятся коды товаров и количества их единиц.
5. Полная информация о заказе сохраняется в файле с заданным именем.

Создать методы, которые осуществляют сериализацию/десериализацию объекта типа База данных товаров. Формат выбрать самостоятельно.

Лабораторная работа №10. Интерфейсы, сортировка с компаратором

Задание:

Дан текстовый файл, содержащий данные о продуктах на складе и их описания, например:

```
3кг Апельсины

10л Квас

100л Вода

3780г Шоколад

10т Бананы

13кг Мангал
```

1. Определите класс с тремя закрытыми полями:
   - Количество в кг (вещественное число);
   - Исходное представление количества (строка);
   - Название (строка).

2. Реализуйте конструктор, принимающий на вход два *строковых* значения: количество и название товара. Конструктор должен генерировать исключение, если количество является некорректным (меньше нуля). Добавьте свойства для преобразования количества в кг. В основной программе загрузите все температурные данные из исходного файла в список `List< >`.

3. Попытайтесь вызвать метод *Sort* для загруженного ранее списка температур. Возникающее при этом исключение свидетельствует о невозможности выполнять сравнение объектов произвольного класса. Чтобы это стало возможным, необходимо, например, *реализовать в классе интерфейс* `IComparable<T>`. Для этого:
   - измените заголовок класса на следующий

     ```
     class  ‘Название’: IComparable<’Название’>
     ```

   - Необходимости реализовать метод сравнения. Метод сравнения должен возвращать отрицательное число, если объект, для которого вызывается метод, *меньше* объекта, переданного в качестве параметра, `0` — если оба объекта *равны*, и положительное число — если исходный объект *больше* — реализуйте этот метод;
4. Убедитесь, что метод Sort работает и сортирует список.

Лабораторная работа №11. Интерфейсы, сортировка с компаратором

Задание:

1. Создать класс Point - точка на плоскости с вещественными координатами x, y. Создать конструктор, ToString() и свойства для доступа к координатам точки.
2. Создайте метод, который генерирует набор (LIst<Point>) случайно расположенных точек в квадрате [0,1]x[0,1].
3. Используя интерфейс IComparer, выведите все точки, упорядочивая их следующими способами:
    - по удалению от начала координат (сначала выводится ближайшая к началу координат, порядок равноудалённых точек не важен);
    - по удалению от оси абсцисс (сначала выводится ближайшая к оси абсцисс, порядок равноудалённых точек не важен);
    - по удалению от оси ординат (сначала выводится ближайшая к оси ординат, порядок равноудалённых точек не важен);
    - по удалению от диагонали первой и третьей четвертей (прямая y=x, порядок равноудалённых точек не важен).

Лабораторная работа №12. Творческий проект

Задание: Разработать программу для игры в крестики-нолики.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Документация C# - URL: https://docs.microsoft.com/ru-ru/dotnet/csharp/