МЕЖГОСУДАРСТВЕННОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ
ВЫСШЕГО ОБРАЗОВАНИЯ
«БЕЛОРУССКО-РОССИЙСКИЙ УНИВЕРСИТЕТ»

Кафедра «Программное обеспечение информационных технологий»

ТЕХНОЛОГИИ ПРОГРАММИРОВАНИЯ

Методические рекомендации к лабораторным работам для студентов специальности 6-05-0611-04 «Электронная экономика» дневной и заочной форм обучения

Могилев 2025

УДК 004.42
ББК 32.973-018
Т38

Рекомендовано к изданию
учебно-методическим отделом
Белорусско-Российского университета

Одобрено кафедрой «Программное обеспечение информационных технологий» «15» июня 2025 г., протокол № 9

Составители: канд. техн. наук, доц. С. К. Крутолевич;
канд. техн. наук, доц. Н. Н. Горбатенко

Рецензент канд. техн. наук, доц. И. В. Лесковец

Методические рекомендации разработаны на основе рабочей программы по дисциплине «Технологии программирования» для студентов специальности 6-05-0611-04 «Электронная экономика» дневной и заочной форм обучения. Предназначены для использования при выполнении лабораторных работ.

Учебное издание

ТЕХНОЛОГИИ ПРОГРАММИРОВАНИЯ

Ответственный за выпуск	В. В. Кутузов
Корректор	А. А. Подошевко
Компьютерная верстка	М. М. Дударева

Подписано в печать 22.09.2025. Формат 60×84/16. Бумага офсетная. Гарнитура Таймс. Печать трафаретная. Усл. печ. л. 2,79. Уч.-изд. л. 2,88. Тираж 21 экз. Заказ № 696.

Издатель и полиграфическое исполнение:
Межгосударственное образовательное учреждение высшего образования
«Белорусско-Российский университет».
Свидетельство о государственной регистрации издателя,
изготовителя, распространителя печатных изданий
№ 1/156 от 07.03.2019.
Пр-т Мира, 43, 212022, г. Могилев.

Содержание

Введение

Целью преподавания дисциплины «Технологии программирования» является изучение и практическое освоение общих принципов и современных методов программирования с использованием объектно-ориентированной парадигмы программирования.

Лабораторные работы по дисциплине проводятся с целью закрепления теоретических знаний, полученных на лекциях, путем их практического применения. В рамках лабораторных работ студенты учатся применять теоретические знания для решения практических задач, что способствует лучшему усвоению материала и формированию профессиональных компетенций.

Последовательность выполнения лабораторных работ.

1 Ознакомиться с целью предстоящей работы.

2 Изучить вопросы пункта «Теоретические сведения», используя указанные литературные источники и конспект лекций.

3 Ответить на контрольные вопросы к лабораторной работе.

4 Составить план решения задачи индивидуального задания.

5 Набрать текст программы решения задачи в среде Visual Studio.NET.

6 Отладить программу и убедиться в ее правильной работе.

7 Подготовить отчёт по лабораторной работе.

8 Защитить лабораторную работу. Защита включает в себя демонстрацию работы программы преподавателю и ответы на его вопросы по теме лабораторной работы в объёме методических рекомендаций. После защиты лабораторной работы преподаватель ставит на титульном листе свою подпись и дату. Только после этого лабораторная работа считается полностью выполненной и студент может приступить к выполнению следующей работы.

Отчёт по лабораторной работе оформляется на листах формата А4 и должен содержать следующее:

– титульный лист;

– тема и цель работы;

– постановка задачи: текст задания согласно варианту;

– краткие теоретические сведения: основные понятия и методы, использованные при выполнении работы;

– листинг программы: полный код программы с подробными комментариями;

– результаты тестирования: скриншоты работы программы с различными входными данными, демонстрирующие ее корректность;

– выводы: краткое описание полученных навыков и возникших при выполнении работы трудностей.

1 Лабораторная работа № 1. Простейшие классы. Инкапсуляция. Свойства

Цель работы: получение практических навыков разработки и использования пользовательского типа данных класс в объектно-ориентированном программировании (ООП).

Теоретические сведения

1 Объектно-ориентированное программирование [1, с. 11–13].
2 Классы: основные понятия [1, с. 100–125].

Контрольные вопросы

1 Что такое класс и объект класса?
2 Какое отличие между описанием класса и объектом класса?
3 Что объявляется в классе?
4 Какие элементы языка C# могут относиться к функциям-членам класса?
5 Какая общая форма описания класса?
6 Какие существуют типы (модификаторы) доступа к членам класса?
7 Может ли отсутствовать тип (модификатор) доступа при описании членов класса?
8 Как создать объект (экземпляр класса)?
9 К каким типам данных принадлежат классы: к типам значения или к ссылочным типам?
10 Какие особенности использования ключевого слова this в классе?
11 Что представляет собой конструктор? Для чего он используется?
12 Какие бывают конструкторы?
13 Для чего предназначены свойства?
14 Какие бывают свойства?
15 Дайте определение принципа полиморфизма в ООП?
16 С помощью каких средств программирования достигается реализация принципа полиморфизма?

Индивидуальное задание

Создайте проект, в котором опишите класс для решения задачи вашего варианта.

Указания – разрабатываемый класс должен содержать следующие элементы: скрытые поля, конструкторы без параметров и с параметрами (имена некоторых полей должны совпадать с идентификаторами параметров), свойства, метод вывода полей и указанный в таблице 1.1 метод.

Составьте тестирующую программу с выдачей результатов. В программе

должна выполняться проверка всех разработанных элементов класса, вывод состояния объекта.

Таблица 1.1 – Варианты заданий

Вариант	Класс	Метод
1	Сотрудник (поля: имя, минимальная зарплата – p)	Доход: $k \cdot p$, где k – повышающий коэффициент
2	Квартира (поля: номер, стоимость 1 м^2, площадь)	Стоимость квартиры
3	Агрополе (поля: название, вес посеянных семян на единицу площади – r, площадь – S, га)	Урожай: $k \cdot r$, где k – коэффициент, зависящий от культуры
4	Стол (поля: название, площадь – S, см)	Стоимость: $S^2/3 + 500000$
5	Автобус (поля: количество пассажиров, стоимость билета)	Выручка
6	Транспортное средство (поля: название, расстояние, цена за 1 км)	Стоимость проезда
7	Пособие (поля: повышающий коэффициент – k, минимальное пособие – r)	Размер пособия: $k \cdot r$
8	Телефон (поля: марка, количество функций – k)	Стоимость: $40 \cdot \ln(k)$
9	Автомобиль (поля: марка, расход горючего на 100 км – N, расстояние – R, км)	Объем горючего: $N \cdot R$
10	Одежда (поля: модель, ширина ткани – H, м, норма расхода – L, м)	Расход ткани: $(2 + H) \cdot L$
11	Постройка (поля: название, высота здания – H, м)	Высота фундамента: $0{,}03 \cdot H$
12	Аудитория (поля: номер, площадь – S, м^2)	Количество мест: $S/1{,}2$

Пример выполнения индивидуального задания

Класс – частный дом. Поля класса: тип дома (кирпичный, блочный, дом из бруса), общая площадь дома – S, м^2, длина ленточного фундамента – L. Метод – вычислить стоимость строительства дома по формуле $300 \cdot S + 200 \cdot L$.

Листинг 1. Код класса House (дом):

```
using System;

namespace Laba1
{
   class House
   {
      // Поля класса
```

```
// Тип частного дома
private string typeHouse;
// Площадь частного дома
private double square;
// Длина ленточного фундамента
private double lengthFoundation;

//Свойство для работы с полем typeHouse
public string TypeHouse
{
   set
   {
      typeHouse = value;
   }
   get
   {
      return typeHouse;
   }
}

//Свойство для работы с полем square
public double Square
{
   set
   {
      if (value > 0)
         square = value;
      else
         square = 0;
   }
   get
   {
      return square;
   }
}

//Свойство для работы с полем lengthFoundation
public double LengthFoundation
{
   set
   {
      if (value > 0)
         lengthFoundation = value;
      else
         lengthFoundation = 0;
   }
   get
   {
      return lengthFoundation;
   }
}

// Конструктор по умолчанию
public House()
{
   typeHouse = "Кирпичный";
   square = 100;
   lengthFoundation = 40;
}

// Конструктор с параметрами
public House(string typeHouse, double square, double lengthFoundation)
```

```
        {
            this.typeHouse = typeHouse;
            this.square = square;
            this.lengthFoundation = lengthFoundation;
        }

        // Метод вычисления стоимости дома
        public double CostHouse()
        {
            return 300 * square + 200 * lengthFoundation;
        }

        // Метод вывода полей класса
        public void InfoHouse()
        {
            Console.WriteLine($"Тип дома: {TypeHouse}. Площадь дома: {Square}. Длина фундамента {LengthFoundation}");
        }

    }
}
```

Листинг 2. Код класса Program:

```
using System;

namespace Laba1
{
    class Program
    {
        static void Main(string[] args)
        {
            // Создаем экземпляр класса House, используя конструктор по умолчанию
            House house1 = new House();
            // Выводим информацию о доме
            house1.InfoHouse();
            // Вычисляем стоимость дома
            Console.WriteLine($"Стоимость дома: {house1.CostHouse()}");
            // Демонстрация использования свойств
            Console.WriteLine($"Исходное значение площади дома: {house1.Square}");
            // Вызывается метод установки свойства
            house1.Square = 200;
            Console.WriteLine($"Новое значение площади дома: {house1.Square}");
            // Создаем экземпляр класса House, используя конструктор с параметрами
            House house2 = new House("Блочный", 150, 70);
            // Выводим информацию о доме
            house2.InfoHouse();
            // Стоимость дома
            Console.WriteLine($"Стоимость дома: {house2.CostHouse()}");
            Console.ReadKey();
        }
    }
}
```

2 Лабораторная работа № 2. Массивы. Индексаторы

Цель работы: получение практических навыков разработки классов с использованием массивов и индексаторов.

Теоретические сведения

1 Массивы [1, с. 126–133].
2 Массивы объектов [1, с. 138–139].
3 Индексаторы [1, с. 157–161].

Контрольные вопросы

1 Для чего предназначены индексаторы?
2 Как определяется базовый тип индексатора?
3 Что записывается в качестве имени индексатора?
4 Что содержит список параметров индексатора?
5 Можно ли в классе объявить более одного индексатора? Если да, то при каком условии?
6 Может ли аргумент индексатора иметь тип, отличный от int?

Индивидуальные задания

Создайте проект, в котором опишите класс для решения задачи вашего варианта (таблица 2.1).

Класс должен содержать закрытое поле двумерного массива, конструктор без параметров, конструктор с параметрами, свойства, индексатор, методы (ввода, вывода, обработки массива). Обработку массива в соответствии с заданием варианта осуществлять в одном методе, исходные данные и результаты работы метода передавать параметрами. В программе должны проверяться все элементы разработанного класса.

Таблица 2.1 – Варианты заданий

Вариант	Тип массива	Размерность	Метод
1	Вещественный	$N \times N$	Найти произведение элементов массива, расположенных между максимальным и минимальным элементами
2	Целочисленный	$N \times M$	Найти произведение элементов массива с четными номерами
3	Целочисленный	$N \times M$	Найти сумму элементов массива, расположенных между максимальным по модулю и минимальным по модулю элементами

Окончание таблицы 2.1

Вариант	Тип массива	Размерность	Метод
4	Вещественный	*N*×*M*	Найти сумму положительных элементов
5	Целочисленный	*N*×*N*	Найти сумму элементов массива с нечетными номерами
6	Вещественный	*N*×*N*	Найти сумму элементов массива, расположенных между первым и вторым нулевым элементами
7	Вещественный	*N*×*N*	Найти произведение элементов массива, расположенных после максимального элемента
8	Целочисленный	*N*×*M*	Найти произведение отрицательных элементов массива
9	Целочисленный	*N*×*N*	Найти сумму модулей элементов массива, расположенных после первого элемента, равного нулю
10	Вещественный	*N*×*M*	Найти сумму элементов массива, расположенных после первого положительного элемента
11	Целочисленный	*N*×*M*	Найти сумму модулей элементов массива, расположенных после первого элемента, равного *s*
12	Вещественный	*N*×*M*	Найти в заданном своим номером столбце максимальный по модулю элемент

Пример выполнения индивидуального задания

Создать проект, в котором описать класс для решения следующей задачи: найти количество элементов массива, лежащих в диапазоне от A до B. Тип массива целочисленный. Размерность массива *N*×*M*.

В листингах 1 и 2 показан текст программы решения данной задачи.

Листинг 1. Код класса Massiv:

```
using System;

namespace Lab2_exampleSolver
{
   class Massiv
   {
      // Признак ошибки
      public bool error = false;
      // Закрытое поле двумерного массива
      private int[,] array;
      // Число строк массива
      private int row;
      // Число столбцов массива
      private int column;

      // Свойства
      public int Row
      {
         set
         {
            if (value > 0)
```

```
            row = value;
        else
            error = false;
    }
    get
    {
        return row;
    }
}
public int Column
{
    set
    {
        if (value > 0)
            column = value;
        else
            error = false;
    }
    get
    {
        return column;
    }
}

// Конструктор без параметров
public Massiv( )
{
    Row = 2;
    Column = 3;
    array = new int[Row, Column];
}

// Конструктор с параметрами
public Massiv(int row, int column)
{
    Row = row;
    Column = column;
    array = new int[Row, Column];
}

// Индексатор
public int this[int i, int j]
{
    set
    {
        if ((i >= 0 && i < Row) && (j >= 0 && j < Column))
            array[i, j] = value;
        else
            error = false;
    }
    get
    {
        if ((i >= 0 && i < Row) && (j >= 0 && j < Column))
            return array[i, j];
        else
        {
            error = false;
            return 0;
        }
    }
}

// Метод ввода массива
```

```
        public void InputArray()
        {
            for(int i = 0; i < Row; i++)
                for(int j = 0; j < Column; j++)
                {
                    Console.Write($"Введите array[{i} {j}]: ");
                    array[i,j] = int.Parse(Console.ReadLine());
                }
        }

        // Метод вывода массива
        public void OutputArray()
        {
            for (int i = 0; i < Row; i++)
            {
                for (int j = 0; j < Column; j++)
                    Console.Write($"{array[i, j]} \t");
                Console.WriteLine();
            }
        }

        // Метод обработки массива
        public int ProcessingArray(int[,] arr, int r, int c, int a, int b)
        {
            int number = 0;
            for (int i = 0; i < r; i++)      // Цикл по строкам
                for (int j = 0; j < c; j++)  // Цикл по столбцам
                    if (arr[i, j] >= a && arr[i, j] <= b)
                        number++;
            return number;
        }
    }
}
```

Листинг 2. Код класса Program:

```
using System;

namespace Lab2_exampleSolver
{
    class Program
    {
        static void Main(string[] args)
        {
            // Создание объекта
            Massiv mas = new Massiv(2, 3);

            // Ввод элементов массива
            mas.InputArray();

            // Вывод элементов массива на дисплей
            Console.WriteLine();
            mas.OutputArray();

            // Создаем двумерный массив
            int[,] arr = new int[mas.Row, mas.Column];
            // Используем индексатор для инициализации массива arr
            for (int i = 0; i < mas.Row; i++)
                for (int j = 0; j < mas.Column; j++)
                    arr[i, j] = mas[i,j]; // mas[i,j] - это вызов индексатора

            // Обработка массива arr
```

```
            int a = 10;
            int b = 50;
            int nbr = mas.ProcessingArray(arr, mas.Row, mas.Column, a, b);
            Console.WriteLine($"\nКоличество элементов массива, лежащих в диапазоне от {a} до {b} = {nbr}");

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

3 Лабораторная работа № 3. Перегрузка методов и операций

Цель работы: научиться перегружать методы, бинарные и унарные операторы для пользовательского класса, применять перегруженные операторы для операций над объектами классов.

Теоретические сведения

1 Перегрузка методов [1, с. 152–153].
2 Операции класса [1, с. 161–168].

Контрольные вопросы

1 Что понимается под перегрузкой методов?
2 Какова основная цель перегрузки методов?
3 Что такое перегрузка операций?
4 Зачем нужна перегрузка операций?
5 Какой синтаксис используется в C# для перегрузки операторов?
6 Какие существуют общие правила и ограничения для перегрузки операторов?
7 Чем отличается перегрузка унарного и бинарного оператора?

Индивидуальные задания

Создайте проект, в котором опишите класс для решения задачи вашего варианта. Разрабатываемый класс должен содержать следующие элементы: скрытые поля, свойства, индексатор, конструктор, перегруженная операция. В программе должна выполняться проверка всех разработанных элементов класса.

1 Описать класс для работы с двумерными массивами вещественных чисел. Реализовать возможность выполнения для согласованных массивов комбинированной операции «+=».

2 Описать класс для работы с двумерными массивами вещественных чисел. Реализовать возможность выполнения для согласованных массивов комбинированной операции «-=».

3 Описать класс для работы с двумерным массивом целых чисел. Обеспечить сравнение массивов на равенство (перегрузку операции «==» для поэлементного сравнения).

4 Описать класс для работы с двумерным массивом целых чисел. Реализовать возможность выполнение операции умножения всех элементов массива на

заданное число.

5 Описать класс – матрица размера 2×2. Элементы матрицы числа типа double. Перегрузить оператор «+» так, чтобы он выполнял операцию сложения матриц размера 2×2.

6 Описать класс – матрица размера 2×2. Элементы матрицы числа типа double. Перегрузить оператор «–» так, чтобы он выполнял операцию вычитания матриц размера 2×2.

7 Описать класс – матрица размера 2×2. Элементы матрицы числа типа double. Перегрузить оператор «*» так, чтобы он выполнял операцию умножения матриц размера 2×2 на действительное число типа double.

8 Описать класс – матрица размера 2×2. Элементы матрицы числа типа double. Перегрузить оператор «*» так, чтобы он выполнял операцию поэлементого умножения матриц A и B размера 2×2. Результатом выполнения операции поэлементного умножения матриц A и B является матрица C, определяемая по формуле C(i,j) = A(i,j)*B(i,j).

9 Описать класс – матрица размера 2×2. Элементы матрицы числа типа double. Перегрузить оператор «/» так, чтобы он выполнял операцию поэлементого деления матриц A и B размера 2×2. Результатом выполнения операции поэлементного деления матриц A и B является матрица C, определяемая по формуле C(i,j) = A(i,j) / B(i,j).

10 Описать класс – матрица размера 2×2. Элементы матрицы числа типа double. Перегрузить оператор «>» так, чтобы он выполнял операцию поэлементого сравнения элементов матриц A и B размера 2×2.

11 Описать класс – матрица размера 2×2. Элементы матрицы числа типа double. Перегрузить оператор «<» так, чтобы он выполнял операцию поэлементого сравнения элементов матриц A и B размера 2×2.

12 Описать класс для работы с двумерным массивом строк. Обеспечить перегрузку операции «+» для построчного соединения элементов.

Пример выполнения индивидуального задания

Описать класс для работы с двумерным массивом целых чисел. Реализовать возможность выполнение операции нахождения остатков от деления всех элементов массива на заданное число (перегрузка операции «%»).

Листинг 1. Код класса Massiv:

```
using System;

namespace LbOverloadOperator
{
   class Massiv
   {

      private int[,] massiv;
      private int row, col; // число строк, столбцов матрицы

      //конструктор
      public Massiv(int row, int col)
      {
```

```
            this.row = row;
            this.col = col;
            massiv = new int[row, col];

            for (int i = 0; i < row; i++)
            {
                for (int j = 0; j < col; j++)
                {
                    massiv[i, j] = (i + 10) * (j + 10);
                }
            }
        }

        //Индексатор
        public int this[int i, int j]
        {
            set { massiv[i, j] = value; }
            get { return massiv[i, j]; }
        }

        // Перегрузка операции %
        public static Massiv operator % (Massiv ar, int nbr)
        {
            for (int i = 0; i < ar.massiv.GetLength(0); i++) // Цикл по строкам
            {
                for (int j = 0; j < ar.massiv.GetLength(1); j++) // Цикл по столбцам
                {
                    ar.massiv[i, j] = ar.massiv[i, j] % nbr;
                }
            }
            return ar;
        }
    }
}
```

Листинг 2. Код класса Program:

```
using System;

namespace LbOverloadOperator
{
    class Program
    {
        static void Main()
        {
            int row = 3;
            int col = 3;
            Massiv mas = new Massiv(row, col);
            mas = mas % 3; //Остаток от целочисленного деления матрицы на 3


            for (int i = 0; i < row; i++)
            {
                for (int j = 0; j < col; j++)
                    Console.Write(" {0} ", mas[i, j]);
                Console.WriteLine();
            }
            Console.ReadLine();
        }
    }
}
```

4 Лабораторная работа № 4. Иерархии классов. Наследование

Цель работы: получение практических навыков разработки классов с использованием наследования.

Теоретические сведения

1 Объектно-ориентированное программирование [1, с. 11–13].
2 Наследование [1, с. 172–178].

Контрольные вопросы

1 Дайте определение принципа наследования в ООП?
2 Какие преимущества дает использование наследования?
3 Дайте определение понятиям «базовый класс» (родительский) и «производный класс» (дочерний). Объясните взаимосвязь между этими двумя типами классов.
4 Какой синтаксис используется в C# для объявления наследования?
5 От какого класса неявно наследуются все классы в C#?
6 Поддерживает ли C# множественное наследование классов?
7 Какие члены класса наследуются?
8 Как вызываются конструкторы при создании объекта производного класса?
9 Для чего используется ключевое слово base в конструкторе производного класса?
10 Что представляет собой защищенный доступ?
11 Что такое сокрытие имен при наследовании?
12 Как получить доступ к сокрытому члену базового класса?

Индивидуальные задания

Создайте проект, в котором опишите иерархию классов для решения задачи вашего варианта (таблица 4.1). Дополните базовый класс, который был создан в лабораторной работе № 1, классами-потомками. Производные классы должны содержать необходимые дополнения и изменения, указанные в варианте задания.

Составить тестирующую программу с выдачей результатов. В программе должна выполняться проверка разработанных элементов всей иерархии классов, вывод состояния различных объектов.

Таблица 4.1 – Варианты заданий

Вариант	Потомки	Дополнение и изменение в потомках
1	Менеджер (поле: объем продаж в тоннах)	Доход менеджера изменить в зависимости от объема продаж (если больше H, то увеличить на 1 % от H)
	Инженер (поле: количество разработанных проектов – n)	Доход инженера увеличить на $4{,}8 \cdot n$
2	Квартира в центре (поля: номер этажа, этажность дома)	Увеличить стоимость с учетом надбавки за расположение в центре на 1 % и уменьшить на 1000 $ для квартир на первом и последнем этажах
	Квартира в пригороде (поле: расстояние от центра – r)	Изменить стоимость в зависимости от расстояния: если меньше 10 км, увеличить на 3 %, в противном случае – уменьшить на $0{,}01 \cdot r$
3	Фермерское (поле: количество внесенных удобрений – m)	Найти урожай с учетом увеличения на $0{,}001 \cdot m$ на единицу площади
	Приусадебное (поля: сроки посева – ранний, средний, поздний)	Изменить урожай с учетом сроков (+10 % – для раннего, 5 % – для позднего) и площади посевов (менее 0,01 увеличить на 50 % от S)
4	Письменный (поля: используемый материал, стоимость отделки)	Увеличить стоимость на стоимость отделки
	Обеденный (поле: форма)	Изменить стоимость в зависимости от формы: увеличить для прямоугольной формы на 10 %, овальной – на 20 % при $S < 0{,}5$ м^2 и $S > 2$ м^2
5	Экспресс (поля: средняя скорость – v, марка автобуса)	Изменить выручку с учетом увеличения на $0{,}05 \cdot v$ к цене билета
	Пригородный (поле: расстояние – r)	Изменить выручку с учетом уменьшения на $0{,}01 \cdot r$ к цене билета
6	Самолет (поля: высота – h, км, скорость – v, км/ч)	Увеличить стоимость проезда на $100 \cdot h \cdot v$
	Корабль (поле: количество палуб – k, номер палубы – n)	Увеличить стоимость проезда на палубах 3 и 4 на k^2 %
7	Инвалид (поле: номер группы)	Увеличить пособие для инвалидов 1-й группы на 30 %, 2-й – на 20 %
	Многодетные семьи (поле: количество детей)	Увеличить пособие от 3 до 5 детей на 10 %, больше 5 – на 20 %
8	Сотовый (поля: модель, год производства)	Изменить стоимость для телефонов, выпущенных менее 1 года назад, – на +20 %, более 3 лет – на -60 %
	Стационарный (поля: мобильность – переносной, непереносной)	Увеличить стоимость переносного телефона на 5,7, уменьшить стоимость непереносного телефона на 3,2

Окончание таблицы 4.1

Вариант	Потомки	Дополнения и изменения в потомках
9	Грузовой (поле: грузоподъемность – p, т)	Увеличить объем горючего на величину $M = p \cdot R$
	Легковой (поле: объем двигателя – V, л)	Увеличить объем горючего для объема больше 3 на величину $M = 0{,}005 \cdot V \cdot R$
10	Пальто (поле: размер – V, л)	Увеличить расход ткани для пальто на $(V/6{,}5 + 0{,}5)/10$
	Костюм (поле: рост – H)	Увеличить расход ткани для костюма на $(2 \cdot H + 0{,}3)/8$
11	Офис (поле: количество этажей – N)	Изменить высоту фундамента для офиса с $N > 10$ на $+ 0{,}005 \cdot N$
	Завод (поле: вес – G)	Изменить высоту фундамента для завода на $+ 0{,}000002 \cdot G$
12	Лекционная (поле: количество ярусов – K)	Увеличить количество мест в лекционной аудитории на $2 \cdot K$
	Компьютерная (поле: количество компьютеров – P)	Заменить количество мест в компьютерной аудитории на $P - 1$

Пример выполнения индивидуального задания

Создадим два класса потомка, приняв в качестве базового класса класс из примера выполнения лабораторной работы № 1.

Потомки:

– коттедж (поле: число этажей – N); дополнения и изменения в потомке: увеличить стоимость дома-коттеджа на $N \cdot S$;

– дачный дом (поле: высота потолка – H, м); дополнения и изменения в потомке: увеличить стоимость дачного дома на $C = 15 \cdot H$.

Листинг 1. Код базового класса House:

```
using System;

namespace Laba4
{
   class House
   {
      // Поля класса
      // Тип частного дома
      private string typeHouse;
      // Площадь частного дома
      private double square;
      // Длина ленточного фундамента
      private double lengthFoundation;

      //Свойство для работы с полем typeHouse
      public string TypeHouse
      {
         set
```

```
        {
            typeHouse = value;
        }
        get
        {
            return typeHouse;
        }
    }

    //Свойство для работы с полем square
    public double Square
    {
        set
        {
            if (value > 0)
                square = value;
            else
                square = 0;
        }
        get
        {
            return square;
        }
    }

    //Свойство для работы с полем lengthFoundation
    public double LengthFoundation
    {
        set
        {
            if (value > 0)
                lengthFoundation = value;
            else
                lengthFoundation = 0;
        }
        get
        {
            return lengthFoundation;
        }
    }

    // Конструктор с параметрами
    public House(string typeHouse, double square, double lengthFoundation)
    {
        this.typeHouse = typeHouse;
        this.square = square;
        this.lengthFoundation = lengthFoundation;
    }

    // Метод вычисления стоимости дома
    public double CostHouse()
    {
        return 300 * square + 200 * lengthFoundation;
    }

    // Метод вывода полей класса
    public void InfoHouse()
    {
        Console.WriteLine($"Тип дома: {TypeHouse}.\t Площадь дома: {Square}.\t Длина фундамента {Length-
Foundation}");
    }
  }
}
```

Листинг 2. Код производного класса Dacha:

```
using System;

namespace Laba4
{
   class Dacha : House
   {
      // Высота комнат
      private double roomHeight;

      // Свойство для доступа к закрытому полю roomHeight
      public double RoomHeight
      {
         set
         {
            if (value > 0)
               roomHeight = value;
            else
               roomHeight = 0;
         }
         get
         {
            return roomHeight;
         }
      }

      // Конструктор с параметрами
      public Dacha(double roomHeight, string typeHouse, double square, double lengthFoundation) : base(typeHouse, square, lengthFoundation)
      {
         this.roomHeight = roomHeight;
      }

      // Метод для вывода мнформации о даче
      public new void InfoHouse()
      {
         Console.WriteLine($"Тип дома: {TypeHouse}. Площадь дома: {Square}. Высота комнат: {roomHeight}. Длина фундамента {LengthFoundation}");
      }

      // Метод для вычисления стоимости дачи
      public double PriceDacha()
      {
         return base.CostHouse() + 15*roomHeight;
      }
   }}\
```

Листинг 3. Код производного класса Cottage:

```
using System;

namespace Laba4
{
   class Cottage : House
   {
      // Число этажей коттеджа
      private double numberLevel;

      // Свойство для доступа к закрытому полю numberLevel
      public double NumberLevel
```

```
        {
            set
            {
                if (value > 0)
                    numberLevel = value;
                else
                    numberLevel = 0;
            }
            get
            {
                return numberLevel;
            }
        }

        // Конструктор с параметрами
        public Cottage(double numberLevel, string typeHouse, double square, double lengthFoundation) :
base(typeHouse, square, lengthFoundation)
        {
            this.numberLevel = numberLevel;
        }

        // Метод для вывода мнформации о коттедже
        public new void InfoHouse()
        {
            Console.WriteLine($"Тип дома: {TypeHouse}. Площадь дома: {Square}. Количество этажей:
{NumberLevel}. Длина фундамента {LengthFoundation}");
        }

        // Метод для вычисления стоимости дома-коттеджа
        public double PriceCottage()
        {
            return base.CostHouse() + base.Square * numberLevel;
        }
    }
}
```

Листинг 4. Код класса Program:

```
using System;

namespace Laba4
{
    class Program
    {
        static void Main(string[] args)
        {
            // Создаем экземпляр класса Cottage
            Cottage cottage = new Cottage(2, "коттедж", 150, 60);

            // Выводим информацию о коттедже
            cottage.InfoHouse();
            Console.WriteLine($"Стоимомость коттеджа: {cottage.PriceCottage()}");

            // Создаем экземпляр класса Dacha
            Dacha dacha = new Dacha(2.5, "дача", 50, 30);

            // Выводим информацию о даче
            dacha.InfoHouse();
            Console.WriteLine($"Стоимомость дачи: {dacha.PriceDacha()}");

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

5 Лабораторная работа № 5. Виртуальные методы и полиморфизм

Цель работы: изучить реализацию принципа полиморфизма через использование виртуальных функций при наследовании.

Теоретические сведения

1 Объектно-ориентированное программирование [1, с. 11–13].
2 Виртуальные методы [1, с. 178–179].

Контрольные вопросы

1 Дайте определение принципа полиморфизма в ООП.
2 Какие существуют виды полиморфизма в C#?
3 Что такое виртуальные методы?
4 Объясните назначение ключевого слова override.
5 Каковы правила переопределения виртуального метода в классе-наследнике?
6 В чем разница между переопределением (override) и сокрытием (new) метода?

Индивидуальные задания

Модифицируйте проект из лабораторной работы № 4 следующим образом.
Создайте базовый класс. Опишите методы в базовом классе как виртуальные. Составьте новое приложение с двумя потомками базового класса. Тестирующая программа должна содержать массив ссылок базового класса на объекты базового и производных классов (количество объектов больше или равно 4). В программе должна выполняться проверка всех разработанных элементов.

6 Лабораторная работа № 6. Абстрактные классы

Цель работы: получение навыков разработки программ с использованием абстрактных классов.

Теоретические сведения

Абстрактные классы [1, с. 181–182].

Контрольные вопросы

1 Объясните основное предназначение абстрактных классов в ООП.
2 Для чего используется ключевое слово abstract при объявлении класса?
3 Можно ли создать экземпляр абстрактного класса? Почему?
4 Может ли абстрактный класс содержать конструктор? Если да, то когда и как он вызывается?
5 Может ли абстрактный класс наследоваться от другого класса (в том

числе другого абстрактного)?

6 Как объявляется абстрактный метод и в чем его особенность?

7 Может ли абстрактный метод иметь тело (реализацию)?

8 В каком типе классов могут объявляться абстрактные методы?

9 Какие члены, помимо абстрактных методов, может содержать абстрактный класс?

10 Какие еще члены класса (кроме методов) могут быть объявлены как abstract?

11 Что обязан сделать класс-наследник, если он наследуется от абстрактного класса?

12 Что произойдет, если класс-наследник не реализует все абстрактные методы базового класса?

13 Какое ключевое слово используется в производном классе для реализации абстрактного метода?

Индивидуальные задания

Модифицируйте проект из лабораторной работы № 5 следующим образом.

Базовый класс опишите как абстрактный. Методы базового класса не должны содержать никакого кода, только заголовок. В тестирующей программе создайте массив ссылок абстрактного класса на объекты производных классов (количество объектов больше или равно 4 и размещаются в массиве в произвольном порядке) и реализуйте указанное в варианте задания дополнение. В программе должна выполняться проверка всех разработанных элементов.

Варианты заданий

1 Организовать вычисление суммарной величины дохода сотрудников и максимальное для менеджеров.

2 Рассчитать среднюю стоимость квартир в центре и найти наибольшее расстояние от города до квартир в пригороде.

3 Найти суммарный урожай фермеров и суммарный урожай с приусадебных участков.

4 Найти суммарную стоимость всех столов и наименьшую площадь обеденных столов.

5 Рассчитать общую выручку и найти наибольшее расстояние для пригородных автобусов.

6 Организовать вычисление средней стоимости проезда на самолетах и средней стоимости проезда кораблем.

7 Определить, на какой вид пособий потребуется больше средств.

8 Найти максимальные стоимости сотовых и стационарных телефонов.

9 Организовать вычисление суммарного расхода горючего и средний расход для легковых автомобилей.

10 Организовать вычисление суммарного расхода ткани для каждого вида одежды.

11 Найти максимальную высоту фундаментов для офиса и минимальную для заводов.

12 Найти общее количество мест в лекционных аудиториях и общее число компьютеров.

7 Лабораторная работа № 7. Интерфейсы

Цель работы: получение навыков разработки программ с использованием интерфейсов.

Теоретические сведения

1 Синтаксис интерфейса [1, с. 188–190].
2 Реализация интерфейса [1, с. 190–193].
3 Работа с объектами через интерфейсы. Операции is и as [1, с. 194–195].
4 Интерфейсы и наследование [1, с. 195–198].

Контрольные вопросы

1 Что такое интерфейс в C#? Объясните концепцию «контракта», который предоставляет интерфейс. Почему интерфейсы считаются одним из ключевых механизмов для достижения полиморфизма?

2 С помощью какого ключевого слова объявляется интерфейс?

3 Какие члены может содержать интерфейс (методы, свойства, события, индексаторы)?

4 Может ли интерфейс содержать поля (переменные экземпляра) или конструкторы? Почему?

5 Имеют ли члены интерфейса по умолчанию модификаторы доступа? Какие?

6 Как класс «реализует» или «наследует» интерфейс? Какой синтаксис для этого используется?

7 Что обязан сделать класс, который реализует интерфейс? Что произойдет, если хотя бы один член интерфейса не будет реализован?

8 Может ли класс реализовывать несколько интерфейсов? Если да, приведите пример синтаксиса.

9 Может ли один интерфейс наследоваться от другого (или нескольких других) интерфейсов? Что это дает и в каких случаях используется?

10 Каково основное назначение оператора is? Какой тип данных возвращает оператор is?

11 Каково основное назначение оператора as? Что возвращает оператор as, если приведение типов возможно? Что возвращает оператор as, если приведение типов невозможно?

12 В чем принципиальная разница между is и as? Опишите сценарий, где лучше использовать is и сценарий, где предпочтительнее as.

Индивидуальные задания

Определить два интерфейса Ix и Iy согласно варианту задания (таблица 7.1). В каждом интерфейсе объявить методы F0() и F1() заданной сигнатуры. Создать класс, реализующий интерфейсы Ix и Iy, содержащий конструктор по умолчанию и два публичных поля valueF0 и valueF1. В теле методов F0() и F1() организовать вычисления указанных математических выражений. Результат работы методов F0() и F1() с типом возврата void присваивать полям класса соответственно valueF0 и valueF1.

В тестирующей программе должны выполняться:

– явная и неявная реализация методов интерфейсов;

– вызов методов с явным приведением к типу интерфейса;

– вызов методов для объекта посредством интерфейсной ссылки.

Таблица 7.1 – Варианты заданий

Вариант	Интерфейс	Поле класса	Значение, возвращаемое методами F0 и F1

1	interface Ix
	{
	void F0(double x);
	void F1(double x);
	}
	interface Iy
	{
	void F0(double xt);
	double F1(int x);
	}
		public double value0, value1	F0 интерфейса Ix возвращает $(2 \cdot x + 5)^3$
			F1 интерфейса Ix возвращает $x^2/(1{,}54 + 2 \cdot x)$
			F0 интерфейса Iy возвращает $1 + e^x$
			F1 интерфейса Iy возвращает $1/(x + 6)$

2	interface Ix
	{
	void F0(double x, out double result);
	double F1(double x);
	}
	interface Iy
	{
	void F0(double x, out double result);
	void F1(int x);
	}
		public double value0, value1	F0 интерфейса Ix возвращает $\sin(x)$
			F1 интерфейса Ix возвращает $\cos(x)$
			F0 интерфейса Iy возвращает $\mathrm{tg}(x)$
			F1 интерфейса Iy возвращает $\mathrm{ctg}(x)$

3	interface Ix
	{
	void F0(int x);
	void F1(int x, out int result);
	}
	interface Iy
	{
	void F0(double x);
	void F1(int x, out int result);
	}
		public double value0, value1	F0 интерфейса Ix возвращает $2 \cdot x + 5$
			F1 интерфейса Ix возвращает $2 \cdot x^3$
			F0 интерфейса Iy возвращает $1 + 2 \cdot x$
			F1 интерфейса Ix возвращает $1 + 4 \cdot x + 5 \cdot x^2$

Продолжение таблицы 7.1

Вари-ант	Интерфейс	Поле класса	Значение, возвращаемое методами F0 и F1

4	interface Ix
	{
	void F0(int x);
	int F1(int x);
	}
	interface Iy
	{
	void F0(int x, out int result);
	int F1(int x);
	}
		public double value0, value1	F0 интерфейса Ix возвращает $20 \cdot x + 50$
			F1 интерфейса Ix возвращает x^5
			F0 интерфейса Iy возвращает $x - 10$
			F1 интерфейса Ix возвращает $25 \cdot x - 23$

5	interface Ix
	{
	void F0(double x);
	double F1(int x);
	}
	interface Iy
	{
	void F0(double x, ref double result);
	double F1(int x);
	}
		public double value0, value1	F0 интерфейса Ix возвращает $(2 \cdot x + 23 \cdot x^2 + 1) / 256$
			F1 интерфейса Ix возвращает $(1 + x^2) / (2 \cdot x)$
			F0 интерфейса Iy возвращает $10 \cdot \sin(2 \cdot x)$
			F1 интерфейса Ix возвращает $20 \cdot \cos(4 \cdot x)$

6	interface Ix
	{
	void F0(double x);
	void F1(double x);
	}
	interface Iy
	{
	void F0(double x, ref double result);
	void F1(double x);
	}
		public double value0, value1	F0 интерфейса Ix возвращает $\ln(x)$
			F1 интерфейса Ix возвращает $\ln(1 + x^2)$
			F0 интерфейса Iy возвращает $1 / \sin(2 \cdot x + 10)$
			F1 интерфейса Ix возвращает $\cos(3 \cdot x) / 5$

7	interface Ix
	{
	double F0(double x);
	void F1(int x);
	}
	interface Iy
	{
	double F0(double x);
	double F1(int x);}
		public double value0, value1	F0 интерфейса Ix возвращает $(20 \cdot x^3 + 3 \cdot x^2 + 10) / 2{,}5$
			F1 интерфейса Ix возвращает $(112 + 2 \cdot x^2) / (2 \cdot x + e^x)$
			F0 интерфейса Iy возвращает $\mathrm{tg}(3 \cdot x)$
			F1 интерфейса Ix возвращает $\mathrm{ctg}(6 \cdot x)$

Окончание таблицы 7.1

Вариант	Интерфейс	Поле класса	Значение, возвращаемое методами F0 и F1

8	interface Ix
	{
	void F0(double x, ref double result);
	void F1(int x);
	}
	interface Iy
	{
	void F0(double x, ref double result);
	int F1(int x);
	}
		public double value0, value1	F0 интерфейса Ix возвращает $(2{,}3\cdot x^4 + 54) / 12{,}6$
			F1 интерфейса Ix возвращает $(1 + x^3) / (2\cdot x - 10)$
			F0 интерфейса Iy возвращает $\sin^2(5{,}2\cdot x)$
			F1 интерфейса Ix возвращает $\cos^2(6{,}3\cdot x)$

9	interface Ix
	{
	double F0(double x);
	int F1(int x);
	}
	interface Iy
	{
	double F0(double x);
	void F1(int x);
	}
		public double value0, value1	F0 интерфейса Ix возвращает $\mathrm{tg}(x) + \mathrm{ctg}(x)$
			F1 интерфейса Ix возвращает $\sin(x) + \cos(x)$
			F0 интерфейса Iy возвращает $1/(5+x) + (23 + x)/5$
			F1 интерфейса Ix возвращает $(2{,}1 + x)\cdot\cos(1{,}5\cdot x + 2)/10$

10	interface Ix
	{
	void F0(double x, out double result);
	void F1(int x);
	}
	interface Iy
	{
	void F0(double x, out double result);
	void F1(int x);
	}
		public double value0, value1	F0 интерфейса Ix возвращает $(2{,}3 \cdot x^5 + 1) / (5{,}6 + x^2)$
			F1 интерфейса Ix возвращает $(1+x^3) / (2\cdot x^2 + 5)$
			F0 интерфейса Iy возвращает $1/\sin(3\cdot x)$
			F1 интерфейса Ix возвращает $2/\cos(5\cdot x)$

11	interface Ix
	{
	void F0(double x);
	double F1(int x);
	}
	interface Iy
	{
	void F0(double x);
	void F1(int x);
	}
		public double value0, value1	F0 интерфейса Ix возвращает $234 \cdot x^2 + 1$
			F1 интерфейса Ix возвращает $(126 - x^3) / (2\cdot x + \ln(x))$
			F0 интерфейса Iy возвращает $\sin(x) + \ln(x)$
			F1 интерфейса Ix возвращает $\cos(x) - \ln(x)$

12	interface Ix
	{
	void F0(double x);
	void F1(double x, ref double result);
	
	}
	interface Iy
	{
	double F0(int x);
	void F1(double x, ref double result);
	}
		public double value0, value1	F0 интерфейса Ix возвращает $\ln(x + 1)$
			F1 интерфейса Ix возвращает $\ln(1 + x^2)$
			F0 интерфейса Iy возвращает $\mathrm{tg}^2(x) + \mathrm{ctg}^2(x)$
			F1 интерфейса Ix возвращает $\cos^2(x) + \sin^2(x)$

Пример выполнения индивидуального задания

Исходные данные представлены в таблице 7.2.

Таблица 7.2 – Исходные данные

Интерфейс	Поле класса	Значение, возвращаемое методами F0 и F1

interface Ix
{
void F0(double x);
double F1(int x);
}
interface Iy
{
void F0(double x, out double result);
double F1(int x);
}
	public double value0, value1	F0 интерфейса Ix возвращает $2 \cdot x + 5$
		F1 интерфейса Ix возвращает x^2
		F0 интерфейса Iy возвращает $\sin(x + 1)$
		F1 интерфейса Iy возвращает $\cos(x + 1)$

Листинг 1. Код интерфейса Ix:

```
using System;

namespace Laba7
{
   interface Ix
   {
      void F0(double x);
      double F1(int x);

   }
}
```

Листинг 2. Код интерфейса Iy:

```
using System;

namespace Laba7
{
   interface Iy
   {
      void F0(double x, out double result);
      double F1(int x);
   }
}
```

Листинг 3. Код класса Class1:

```
using System;

namespace Laba7
{
   public class Class1 : Ix, Iy
   {
      // Поле для хранения результата работы метода F0
```

```
        public double valueF0;
        // Поле для хранения результата работы метода F1
        public double valueF1;

        /// <summary>
        /// Конструктор по умолчанию
        /// </summary>
        public Class1()
        {
            valueF0 = 0;
            valueF1 = 0;
        }

        /// <summary>
        /// Неявная реализация метода F0 интерфейса Ix
        /// </summary>
        /// <param name="x">Параметр</param>
        /// <returns></returns>
        public void F0(double x)
        {
            valueF0 = 2 * x + 5;
        }

        /// <summary>
        /// Неявная реализация метода F1 интерфейса Ix
        /// </summary>
        /// <param name="x">Параметр</param>
        /// <returns></returns>
        public double F1(int x)
        {
            return x*x;
        }


        /// <summary>
        /// Неявная реализация метода F0 интерфейса Iy
        /// </summary>
        /// <param name="x">Параметр</param>
        /// <returns></returns>
        public void F0(double x, out double result)
        {
            result = Math.Sin(x + 1);
        }

        /// <summary>
        /// Явная реализация метода F1 интерфейса Iy
        /// </summary>
        /// <param name="x">Параметр</param>
        /// <returns></returns>
        double Iy.F1(int x)
        {
            return Math.Cos(x + 1);
        }
    }
}
```

Листинг 4. Код класса Program:

```
using System;

namespace Laba7
{
```

```
    class Program
    {
        static void Main(string[] args)
        {
            Class1 obj = new Class1();

            // Вызов методов F0() и F1() интерфейса Ix
            obj.F0(5);
            obj.valueF1 = obj.F1(10);
            Console.WriteLine($"\nРезультат работы метода F0() интерфейса Ix = {obj.valueF0}");
            Console.WriteLine($"\nРезультат работы метода F1() интерфейса Ix = {obj.valueF1}");

            // Вызов метода F0() интерфейса Iy
            double result;
            obj.F0(10, out result);
            Console.WriteLine($"\nРезультат работы метода F0() интерфейса Iy = {result}");

            // Вызов метода F1() интерфейса Iy
            result = (obj as Iy).F1(25);
            Console.WriteLine($"\nРезультат работы метода F1() интерфейса Iy = {result}");

            // Создаем интерфейсную ссылку и присваиваем ей объект класса
            Ix inref = obj;
            Console.WriteLine("\nВызов метода F0() интерфейса Ix с помощью интерфейсной ссылки");
            inref.F0(5);
            Console.WriteLine($"\nРезультат работы метода F0() интерфейса Ix = {obj.valueF0}");

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

8 Лабораторная работа № 8. Стандартные интерфейсы среды .NET

Цель работы: получение навыков разработки программ с использованием стандартных интерфейсов .NET.

Теоретические сведения

Стандартные интерфейсы .NET [1, с. 198–211].

Контрольные вопросы

1 Зачем нужны стандартные интерфейсы .NET? Почему не создавать свои собственные интерфейсы для каждой задачи?

2 В чем разница между обобщенными (generic; например, IComparable<T>) и необобщенными (non-generic; например, IComparable) версиями стандартных интерфейсов? Какой вариант предпочтительнее и почему?

3 Интерфейс IComparable<T>. Каково его основное назначение? Какой метод он содержит и что должны означать его возвращаемые значения (< 0; 0; > 0)?

4 Интерфейс IEquatable<T>. Какую проблему он решает по сравнению с

переопределением object.Equals(object obj)? Какой еще важный метод необходимо переопределить при реализации IEquatable<T> и почему?

5 Интерфейс IEnumerable<T>. Каково его единственное предназначение? Какой единственный метод он содержит?

6 Интерфейс IEnumerator<T>. Как он связан с IEnumerable<T>? Опишите назначение его основных членов: свойства Current и метода MoveNext(). Что происходит при первом вызове MoveNext()?

Индивидуальные задания

Выполнить задания, используя для хранения экземпляров разработанных классов стандартную коллекцию List. Во всех классах реализовать интерфейсы IComparable и IComparer. Перегрузить операции отношения для реализации сравнения объектов по указанному полю. Результат вывести на экран. Организовать вывод коллекции.

Варианты заданий

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

2 Составить список студентов, включающий ФИО, курс, группу, дату и результат забега. Вывести в новый список 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, отсортировать по результатам.

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

4 Составить список студентов группы, включив следующие данные: ФИО, номер группы, результаты сдачи трех экзаменов. Вывести в новый список информацию о студентах, успешно сдавших сессию, отсортировав по номеру группы.

5 Составить список вкладчиков банка, включив следующие данные: ФИО, номер счета, сумма, дату открытия счета. Вывести в новый список информацию о тех вкладчиках, которые открыли вклад в текущем году, отсортировав их по сумме вклада.

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

7 Составить инвентарную ведомость игрушек, включив следующие данные: название игрушки, ее стоимость (в рублях), возрастные границы детей, для

которых предназначена игрушка. Вывести в новый список информацию о тех игрушках, которые предназначены для детей от *N* до *M* лет, отсортировав их по стоимости.

8 Составить список студентов группы, включив следующие данные: ФИО, дату рождения, какую школу окончил. Вывести в новый список информацию о студентах, окончивших заданную школу, отсортировав их по году рождения.

9 В библиотеке имеется список книг. Каждая запись этого списка содержит фамилии авторов, название книги, год издания, цена экземпляра. Определить, имеются ли в данном списке книги, в названии которых встречается некоторое ключевое слово (например, «программирование»). Если имеются, то вывести фамилии авторов, название и год издания всех таких книг, отсортировав их по году издания.

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

11 Составить список больных отделения, включив следующие данные: ФИО, болезнь, дата поступления, рабочий стаж. Вывести в новый список информацию о больных, находящихся на лечении больше недели, отсортировав их по ФИО.

12 В пресс-центре выставки программных средств хранятся данные о каждом экспонате: название, автор, количество заявок на него. Вывести в новый список экспонаты, получившие больше трех заявок, отсортировав по числу заявок.

Пример выполнения индивидуального задания

Создать класс Student, описывающий работу со студентами. Пусть у него будут поля: fio – фамилия студента; ochenka – средняя оценка. В программе main() создать массив типа List и затем отсортировать этот массив. Сортировку сделать по фамилиям в алфавитном порядке, в случае одинаковых фамилий – по возрастанию средних оценок.

Программа решения задачи:

```
using System;
using System.Collections.Generic;

// Пространство имен для нашего проекта
namespace StudentSorterApp
{
    /// <summary>
    /// Класс, описывающий студента.
    /// </summary>
    public class Student
    {
        /// <summary>
        /// Фамилия, имя и отчество студента.
        /// </summary>
        public string Fio { get; set; }

        /// <summary>
        /// Средняя оценка студента.
```

```
        /// </summary>
        public double Ochenka { get; set; }

        /// <summary>
        /// Конструктор для удобного создания объекта студента.
        /// </summary>
        /// <param name="fio">ФИО студента.</param>
        /// <param name="ochenka">Средняя оценка.</param>
        public Student(string fio, double ochenka)
        {
            Fio = fio;
            Ochenka = ochenka;
        }

        /// <summary>
        /// Переопределяем стандартный метод ToString() для красивого вывода в консоль.
        /// Этот метод будет автоматически вызываться, когда мы пытаемся преобразовать объект в строку,
        /// например, в Console.WriteLine(student).
        /// </summary>
        /// <returns>Строковое представление объекта Student.</returns>
        public override string ToString()
        {
            // Используем форматирование строк для аккуратного вывода
            return $"Студент: {Fio,-20} | Средняя оценка: {Ochenka:F2}";
        }
    }

    /// <summary>
    /// Основной класс программы.
    /// </summary>

    class Program
    {
        static void Main(string[] args)
        {
            // Устанавливаем кодировку консоли для корректного отображения кириллицы
            Console.OutputEncoding = System.Text.Encoding.UTF8;

            // 1. Создаем коллекцию типа List<Student>
            var students = new List<Student>
            {
                new Student("Петров П.П.", 4.9),
                new Student("Иванов И.И.", 4.5),
                new Student("Сидоров С.С.", 3.8),
                new Student("Иванов И.И.", 4.2), // Специально добавляем однофамильца для проверки сорти-
ровки
                new Student("Алексеев А.А.", 5.0),
                new Student("Иванов И.И.", 4.8)  // И еще одного для более сложного случая
            };

            // 2. Выводим исходный список студентов в консоль
            Console.WriteLine("--- Исходный список студентов ---");
            foreach (var student in students)
            {
                Console.WriteLine(student);
            }

            // 3. Сортируем список
            // Мы используем метод Sort, которому передаем лямбда-выражение в качестве правила сравнения.
            // Это выражение принимает два объекта Student (s1 и s2) и должно вернуть:
            //   - отрицательное число, если s1 должен идти раньше s2
            //   - ноль, если их порядок не важен
            //   - положительное число, если s1 должен идти после s2
```

```
            students.Sort((s1, s2) =>
            {
                // Сначала сравниваем фамилии в алфавитном порядке
                int fioComparison = string.Compare(s1.Fio, s2.Fio);

                // Если фамилии не равны (результат сравнения не 0), то возвращаем этот результат.
                // Порядок определен.
                if (fioComparison != 0)
                {
                    return fioComparison;
                }
                // Если же фамилии одинаковы (fioComparison == 0), то переходим к сравнению по оценкам.
                // Сравниваем оценки по возрастанию. Метод CompareTo идеально для этого подходит.
                else
                {
                    return s1.Ochenka.CompareTo(s2.Ochenka);
                }
            });

            // 4. Выводим отсортированный список студентов
            Console.WriteLine("\n--- Отсортированный список студентов ---");
            Console.WriteLine("(Сортировка по ФИО, затем по возрастанию оценки)");
            foreach (var student in students)
            {
                Console.WriteLine(student);
            }

            // Ожидаем нажатия клавиши перед закрытием консоли
            Console.ReadKey();
        }
    }
}
```

9 Лабораторная работа № 9. Символы и строки

Цель работы: изучение работы со строками и символами в языке программирования C#.

Теоретические сведения

Символы и строки [1, с. 139–148].

Контрольные вопросы

1 Перечислите способы создания объектов типа string.
2 Перечислите операции над строками.
3 В чём особенность операции индексирования для строк?
4 В чём отличия и в чём сходство строк и массивов типа char[ ]?
5 В каких случаях метод ToString() вызывается неявно?
6 Как выполняется сравнение строк?
7 Как выполняется метод Join()?
8 Как выполняется метод Split()?
9 Перечислите особенности использования строковых объектов класса System.String.

10 В чем заключается различие между строковыми объектами классов String и StringBuilder?

11 Объясните назначение класса StringBuilder.

12 В каких случаях рекомендуется использовать строковые объекты класса StringBuilder?

13 В чем заключается главная особенность строковых объектов класса StringBuilder?

14 Перечислите основные методы и свойства класса StringBuilder.

Индивидуальные задания

Задание 1

Используя тип string разработать программу, которая для заданной строки *s*:

1) подсчитывает общее число вхождений символов *x* и *y*;

2) определяет, какой из двух заданных символов встречается чаще в строке;

3) выводит на экран символы, которые наиболее часто встречаются в строке;

4) выводит на экран символы, которые встречаются в строке только один раз;

5) определяет, является ли строка палиндромом;

6) определяет, упорядочены ли по алфавиту символы строки;

7) подсчитывает количество цифр в строке;

8) подсчитывает сумму всех содержащихся в ней цифр;

9) выводит на экран последовательность символов, расположенных до первого двоеточия;

10) выводит на экран последовательность символов, расположенных после последнего двоеточия;

11) выводит только те слова сообщения, которые содержат не более чем *n* букв;

12) выводит только те слова сообщения, которые содержат хотя бы одну цифру.

Задание 2

Используя класс StringBuilder разработать программу, которая для заданной строки *s*:

1) вставляет символ x после каждого вхождения символа *y*;

2) удваивает каждое вхождение заданной подстроки *x*;

3) удаляет все символы *x*;

4) удаляет все подстроки substr;

5) заменяет все вхождения подстроки str1 на подстроку str2 (при этом str1 может являться частью str2);

6) заменяет все группы стоящих рядом точек на многоточие;

7) меняет местами первую букву со второй, третью с четвертой и т. д;

8) меняет местами первую букву с последней, вторую с предпоследней и т. д;

9) удаляет из строки все подстроки, состоящие из цифр;

10) удаляет из строки самую длинную подстроку, состоящую из повторяющегося символа.

11) удаляет из сообщения все повторяющиеся слова (без учета регистра);

12) подсчитывает сколько раз заданное слово встречается в сообщении.

10 Лабораторная работа № 10. Исключительные ситуации

Цель работы: получение навыков использования механизма обработки исключительных ситуаций в прикладных программах.

Теоретические сведения

Обработка исключительных ситуаций [1, с. 89–95].

Контрольные вопросы

1 Что такое «исключительная ситуация (исключение)»?
2 Какие операторы используются для обработки исключений? Для чего используется ключевое слово try?
3 Проиллюстрируйте обработку исключительной ситуации фрагментом программы на языке C#.
4 Перечислите основные системные исключения.
5 Требуется ли соблюдение соответствия типа перехватываемого исключения типу исключения в обработчике catch?
6 Как реализуется перехват всех исключений?
7 Что происходит, если исключение не обработано?
8 Как выполняется оператор catch без параметров?
9 Может ли быть в программе блок try без блока catch?
10 Куда передается управление после обработки исключительной ситуации?
11 Как осуществляется явная генерация исключения? Какой оператор используется для явной генерации исключения?
12 Как можно повторно перехватить исключение в программе?
13 Можно ли создавать собственные классы исключений для обработки ошибок в программе пользователя?
14 Какой системный класс является базовым для создания классов исключений пользователя?
15 Как обеспечивается обращение к свойствам и методам системных классов исключений?
16 Получают ли создаваемые пользователем классы исключений доступ к свойствам и методам, определенным в классе Exception?
17 Как завершится программа, если возникло исключение, для которого нет соответствующего обработчика?

Индивидуальные задания

Во всех заданиях реализуемые функции или методы должны генерировать подходящие исключения. Обработку исключений нужно выполнять главной функцией, которая должна демонстрировать обработку всех перехватываемых исключений.

Функции, реализуемые в заданиях, обязаны выполнять проверку передаваемых параметров и генерировать исключение в случае ошибочных. Все функции реализуются в четырех вариантах:

1) без спецификации исключений;
2) со спецификацией throw();
3) с конкретной спецификацией с подходящим стандартным исключением;
4) спецификация с собственным реализованным исключением.

Собственное исключение должно быть реализовано в трех вариантах: как пустой класс, как независимый класс с полями-параметрами функции, как наследник от стандартного исключения с полями. Перехват и обработку исключений должна выполнять главная функция.

Варианты заданий

1 Функция вычисляет площадь треугольника по трем сторонам $S = \sqrt{p(p-a)(p-b)(p-c)}$, где $p = (a+b+c) / 2$.

2 Функция вычисляет корень линейного уравнения $ax + b = 0$.

3 Функция вычисляет периметр треугольника.

4 Функция переводит часы и минуты в секунды.

5 Функция вычисляет корень квадратного уравнения $ax + bx + c = 0$.

6 Функция вычисляет сумму геометрической прогрессии $S_n = (a_0 - a_n r) / (1 - r)$.

7 Функция выполняет деление комплексных чисел A и B. Комплексные числа представлены структурой-парой.

8 Функция вычисляет целую часть неправильной дроби, представленной числителем и знаменателем – целыми числами.

9 Функция переводит комплексное число $z = x + i \cdot y$ из алгебраической формы в тригонометрическую $z = radius(\cos(angle) + i \cdot \sin(angle))$. Комплексное число представлено структурой-парой. Преобразованное число тоже представляется структурой-парой (*radius*, *angle*): $radius = \sqrt{x^2 + y^2}$; $angle = \arcsin\frac{y}{x}$.

10 Функция вычисляет разность между двумя датами в днях. Даты представлены структурой с тремя полями: год, месяц, день.

11 Функция вычисляет продолжительность телефонного разговора в минутах, принимая время начала и окончания. Время представлено структурой с тремя полями: час, минута, секунда. Неполная минута считается за полную.

12 Функция вычисляет углы прямоугольного треугольника. В качестве параметров передаются катеты A и B. (Синус угла *Al*, противолежащего катету A, вычисляется по формуле $\sin(Al) = a / c$, где c – гипотенуза треугольника.)

11 Лабораторная работа № 11. Делегаты. События

Цель работы: овладение основными приемами событийного программирования и их программной реализацией.

Теоретические сведения

1 Делегаты [1, с. 220–232].
2 События [1, с. 232–237].

Контрольные вопросы

1 Что такое «делегат»?
2 В чем состоит преимущество использования делегатов?
3 Когда осуществляется выбор вызываемого метода в случае использования делегата?
4 Что является значением делегата?
5 Какое ключевое слово используется для описания делегата?
6 Проиллюстрируйте использование делегата фрагментом программы на языке C#.
7 Должна ли сигнатура вызываемого метода соответствовать сигнатуре делегата?
8 Возможен ли вызов метода в том случае, если его сигнатура не соответствует сигнатуре делегата?
9 Что такое «многоадресный делегат»?
10 Как можно создать последовательность методов для многоадресных делегатов?
11 Какие операторы используются для создания последовательности методов для многоадресных делегатов?
12 Как можно удалить элемент из последовательности методов для многоадресных делегатов?
13 Каким должен быть тип возвращаемого значения для многоадресных делегатов?
14 Как можно подписаться на событие?
15 Каким образом вызывается метод, подписавшийся на событие? Он всегда работает пока ждёт события?
16 Что такое событие в языке C# ?
17 Объясните синтаксис оператора посылки сообщения.
18 Приведите формат объявления события.
19 Что такое переменная события?
20 Что определяет делегат, указанный в объявлении события?
21 Какие действия предусматривает подписка на события?
22 Назовите этапы работы с событиями.

Индивидуальные задания

Задание 1

Разработать проект использования делегата (таблица 11.1) для:
– вызова разных методов одним экземпляром делегата;
– многоадресной передачи.
В методах предусмотреть вывод результата.

Таблица 11.1 – Варианты заданий

Вариант	Параметр делегата	Метод возвращает		
		Класс 1		Класс 2
		метод_1 (статич.)	метод_2 (экземп.)	
1	double	$\sqrt{w}$	h^2+5	e^p
2	double	x^2	$15/y$	n^3
3	char	Символ, преобразованный в нижний регистр	Цифру 5, если символ – буква	*, если символ – буква
4	string	Строку, удалив два первых символа	Строку, заменив первый символ символом «–»	Строку, удалив два последних символа
5	char, bool	Определяет, является ли символ (код символа) цифрой	Определяет, содержится ли значение кода символа в диапазоне кодов ASCII	Определяет, является ли символ знаком препинания
6	int, string	Удвоенную строку с удаленным k-м символом	Строку, заменив s-й символ знаком «+»	Подстроку, начиная с h-й позиции
7	double	$\sin(r)$	$n + 2$	$2/y \cdot \ln(y)$
8	int, int	$n \cdot u^2$	$2 \cdot w - t$	$\lvert x\rvert - k$
9	double	$1/e^p$	$x/10$	$w - 1010 \cdot w$
10	char, int	Если символ является цифрой, найти его числовое значение, иначе, его код	Вернуть код символа	Если символ является буквой «x», вывести число 0
11	int, float	$7 \cdot m/\ln(2)$	$7 \cdot i \cdot 2{,}6$	$\lvert k - 10\rvert$
12	string	Заменить все буквы в строке на прописные символы	Изменить регистр всех букв в строке	Заменить на пробелы знаки препинания

Задание 2

Создать событие на основе делегата (см. задание 1, таблицу 11.1). В тестирующем классе организовать цепочку вызовов.

12 Лабораторная работа № 12. Файлы

Цель работы: получение навыков разработки программ с использованием файлов.

Теоретические сведения

Работа с файлами [1, с. 246–263].

Контрольные вопросы

1 Что такое поток данных?
2 Для чего предназначен механизм потоков данных в языке C#?
3 Какие классы иерархии потоков ввода вы знаете?
4 Какие стандартные классы используются при консольном вводе?
5 Какие стандартные классы используются при файловом вводе?
6 В чем отличие текстового набора данных от бинарного набора?
7 Объясните механизм консольного ввода.
8 Какие классы иерархии потоков вывода вы знаете?
9 Какие стандартные классы используются при консольном выводе?
10 Какие стандартные классы используются при файловом выводе?
11 Какая передача данных выполняется быстрее: в текстовый набор данных или в бинарный набор данных? Объясните почему.
12 Стандартные классы, объекты файлового ввода/вывода.
13 Объясните механизм файлового ввода.
14 Объясните механизм консольного вывода.
15 Объясните механизм файлового вывода.
16 Какие операторы используются для открытия текстового потока данных?
17 Какие операторы используются для открытия бинарного потока данных?

Индивидуальные задания

Задание 1

Записать в текстовый файл результат расчета функции $f(y)$ (таблица 12.1). Результат должен быть записан в виде двух столбцов – аргумента и значения функции от данного аргумента. Начало и конец диапазона, имя файла, а также шаг расчета вводить с клавиатуры.

Таблица 12.1 – Варианты заданий

Вариант	Функция
1	$f(y) = y \cdot y$
2	$f(y) = y \cdot y \cdot y$
3	$f(y) = \cos(y)$
4	$f(y) = \sin(y)$
5	$f(y) = \sin(y) \cdot \cos(y)$
6	$f(y) = \sin(y) \cdot y$
7	$f(y) = \cos(y) \cdot y$
8	$f(y) = \sin(y) \cdot \cos(y) \cdot y$
9	$f(y) = \sin(y) \cdot y \cdot y$
10	$f(y) = \cos(y) \cdot y \cdot y$

Задание 2
Считать файл, вывести на экран среднее арифметическое.

13 Лабораторная работа № 13. Разработка Windows-приложений

Цель работы: получение навыков разработки программ с помощью технологии Windows Forms.

Теоретические сведения

Введение в программирование под Windows [1, с. 311–343].

Контрольные вопросы

1 Что понимают под термином «событийно-управляемое программирование»?
2 Назовите основные структурные элементы Windows-приложения.
3 Опишите процесс создания Windows-приложения с графическим интерфейсом на основе Windows Forms.
4 Что такое обработчик события?
5 Как создаются обработчики событий?
6 Опишите синтаксис обработчика событий.
7 Какие параметры передаются в обработчик события?
8 Перечислите основные свойства стандартных визуальных элементов управления.
9 Каким образом можно добавить элемент управления на форму?
10 Объясните назначение класса Form.
11 Назовите основные свойства, методы, события класса Form.

12 Чем диалоговое окно отличается от обычной формы?

13 Какие виды диалоговых окон вам известны?

14 Объясните последовательность действий, которые надо совершить для отображения диалогового окна.

15 Для чего предназначен класс Application?

16 Перечислите основные элементы класса Application.

Индивидуальные задания

Задание 1

Работа с формой (рисунок 13.1).

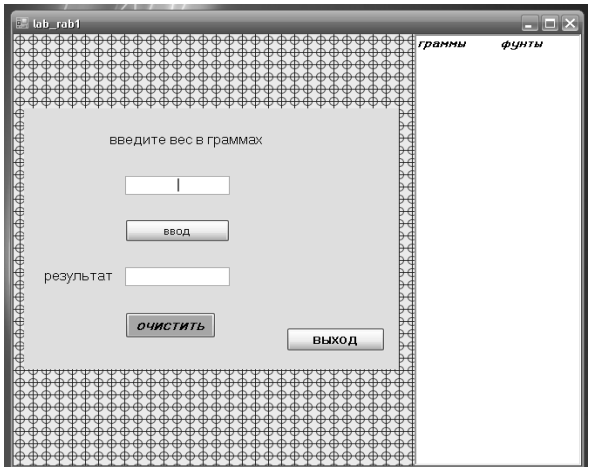

Рисунок 13.1 – Вид окна формы

Для создания приложения, изображенного на рисунке 13.1, выполните следующие действия.

1 Создайте новый проект: File/New/Project, либо New в окне Start Page.

2 В окне New project в левой части выбрать Visual C# Projects, в правой – пункт Windows Forms Application.

3 В поле Name введите имя проекта, в поле Location – место его сохранения на диске.

4 Просмотрите свойства формы, представленные на странице Properties.

5 Измените свойство Text с Forml на «Лабораторная работа 13».

6 Обратите внимание на свойство Name. Это свойство определяет имя компонента, под которым компонент будет известен программе.

7 Измените свойство BackGroundImage Forml на какой-либо рисунок. Для этого в окне Select Resource установите переключатель Local Resource, нажмите Import и выберите файл изображения.

8 Поместите на форму компонент Panel. Осуществите прогон пустой программы. В рабочем приложении максимизируйте окно, а затем закройте его.

9 Задайте свойству компонента BackColor Panel1 значение ControlLight.

10 Разместите Panel1 в центре формы.

11 Отбуксируйте сторону компонента Panel, ухватившись за верхний обрам-

ляющий черный квадратик. Обратите внимание на то, что это значение установилось в свойстве Height инспектора объектов. Задайте размер Size 430; 320.

12 Установите на панель Panel1 метку Labei1. Свойству Text придайте значение «Введите значение веса в граммах». В свойстве Font раскройте диалоговое окно настройки шрифта и измените высоту шрифта – 12 пт.

13 Установите на панель Panel1 редактор TextBox1. Очистите свойство Text.

14 Установите на панель кнопку Buttonl. Задайте свойству Text значение «Ввод».

15 Поместите на панель Panel1 метку Label2. Свойству Text придайте значение «Результат». Установите высоту шрифта 12 пт.

16 Установите на панель Panel1 редактор TextBox2. Очистите свойство Text. Установите начертание шрифта – курсив, размер 10 пт.

17 Поместите на форму компонент richTextBox1 (многострочный редактор) и задайте свойству Dock значение Richt. Компонент займет правую часть формы.

18 Свойству многострочного редактора Lines введите текст «Граммы Фунты» через две табуляции.

19 Установите на панель кнопку Button2. Задайте свойству Text значение «Выход». Задайте высоту шрифта 12 пт. Измените свойство UseVisualStyleBackColor на false.

20 Выделите кнопку Buttonl. В инспекторе щелкните по закладке events. Дважды щелкните по правой части строки события OnClick. В ответ C# активизирует окно программы. Событие OnClick возникает в работающей программе при щелчке мышью по кнопке.

21 Используя линейки прокрутки, просмотрите содержимое окна программы.

22 Занесите в программу (обработчик события) следующий код.

После public partial class Form1 : Form введите описания переменных:

```
double x; //значение веса в граммах, вводимое в textBox1
double y; // значение веса в фунтах
int i=1; //количество символов, в строке richTextBox1
```

В методе private void button1_Click(object sender, EventArgs e) между операторными скобками { } введите следующий код:

```
x = Convert.ToDouble(textBox1.Text);
y = x / 400;
textBox2.Text= Convert.ToString(y);
MessageBox.Show("Нажата кнопка buttonl"); richTextBox1.Multiline=true;
richTextBox1.Text += (x+ "\t\t"+Convert.ToString(y) + "\n");
MessageBox.Show("Нажата клавиша " + e.ToString()); richTextBox1.TabIndex = i;
i += (Convert.ToString(y) + (char)13).Length;
```

Пояснения – В первом операторе присваивания содержимое окна редактора преобразуется в вещественное число. Для отражения результата расчета в окне textBox2 используется свойство Text. этого компонента. Свойство Text класса richTextBox добавляет новую строку к имеющемуся в Text набору строк. Добавленная строка отображается на экране. Свойство TabIndex устанавливает номер позиции в тексте richTextBox1.Text. Последний оператор изменяет эту позицию на длину добавленной строки.

23 Создайте обработчик события для кнопки Button2, введя текст Close();

24 Сохраните программу на диске.

25 Выполните программу. Введите в окно редактора любое число, нажмите

на кнопку «Ввод» и вы получите результат в окнах редакторов.

26 Закомментируйте вывод в окна MessageBox.

27 Добавьте на панель кнопку Button3. Задайте свойству Text значение «Очистить». Напишите обработчик события для очистки окон textBox1 и textBox2, а также установки курсора в окно textBox1 (метод Focus()).

Задание 2

Создайте приложение, в соответствии с рисунком 13.2, для вычисления корней квадратного уравнения.

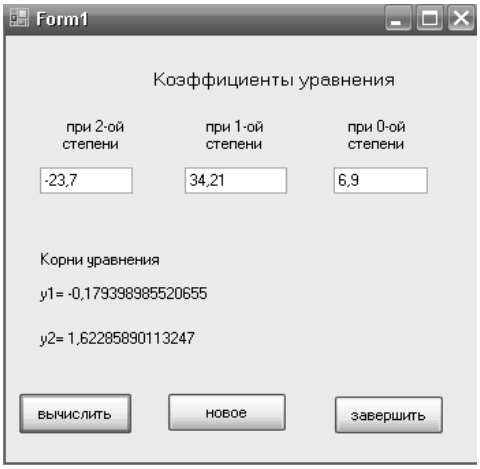

Рисунок 13.2 – Вид окна приложения

Создайте обработчик событий, который:

– предоставит защиту от ввода недопустимых символов в редакторы Editl, Edit2, Edit3;

– будет вычислять корни квадратного уравнения;

– будет подготавливать окно к новому расчету;

– завершает работу приложения.

Задание 3

Использование элементов управления Common Controls и создание диалоговых окон. Создайте приложение (рисунок 13.3) для пересчета из одной системы измерения в другую, используя следующие соотношения: 1 фунт = 400 г; 1 пуд = = 16380 г; 1 унция = 28,35 г; 1 драхм = 1,772 г; 1 гран = 0,0647989 г.

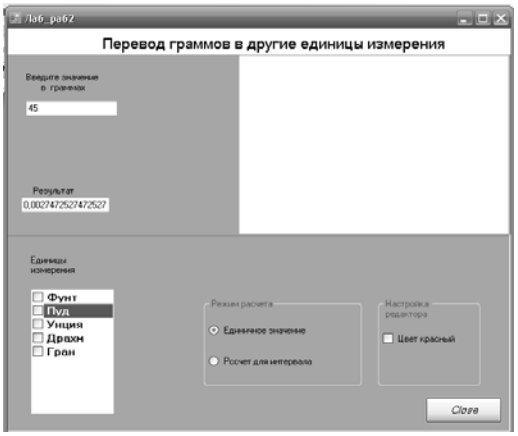

Рисунок 13.3 – Вид окна приложения с элементами управления

Под расчётом для интервала понимается следующее.

При нажатии на радиокнопку «Расчет для интервала» должно появиться три новых текстовых редактора TextBox в одном из которых указывается начальное значение вычисляемой величины, во втором – конечное значение вычисляемой величины, а в третьем – шаг приращения от начального к конечному значению. Результаты вычислений для выбранной единицы измерения должны отражаться в белом правом поле – это компонент richTextBox. Описанные действия достаточно сделать только для одной единицы измерения. При активизации поля «Цвет красный» следует изменить цвет единиц измерения с синего на красный.

Задание 4

Создание Windows-приложений с меню. Разработать приложение, изображенное на рисунке 13.4.

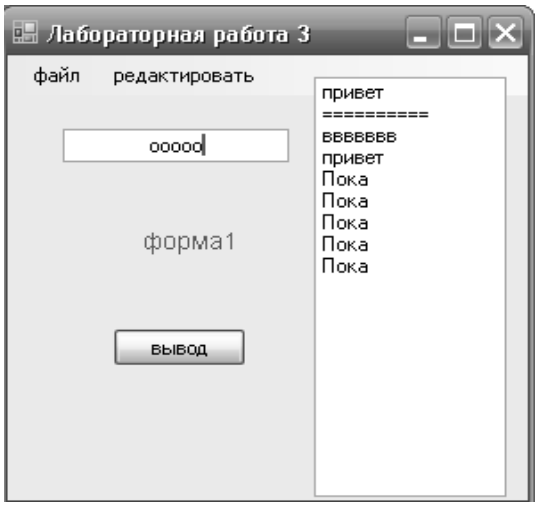

Рисунок 13.4 – Приложение с меню

Меню приложения содержит команды «Файл» и «Редактировать». Команда «Файл» содержит пункты: «Сохранить», «Открыть», «Выход». Команда «Редактировать» содержит пункты «Вставить», «Удалить».

Кнопка «Вывод» выводит на форму сообщение и содержимое редактора TextBox в очередную строку списка ListBox.

При выборе команды «Файл/Сохранить» открывается диалоговое окно для сохранения в файле содержимого списка ListBox.

При выборе команды «Файл/Открыть» открывается диалоговое окно для чтения файла в список ListBox.

Команда «Редактировать/Вставить» вставляет текст из TextBox в ListBox.

Команда «Редактировать/Удалить» удаляет выделенный диапазон записей из ListBox.

Команда «Файл /Выход» закрывает окно.

Список литературы

1 **Павловская, Т. А.** C#. Программирование на языке высокого уровня: учебник / Т. А. Павловская. – СПб.: Питер, 2024. – 432 с.

2 **Шилд, Г.** C# 4.0: полное руководство: пер. с англ. / Г. Шилд. – М.: Вильямс, 2023. – 1056 с.

3 **Потапова, Л. Е.** Объектно-ориентированное программирование на языке C#: метод. рекомендации к выполнению лабораторных работ / Л. Е. Потапова, Т. Г. Алейникова. – Витебск: ВГУ им. П. М. Машерова, 2016. – 50 с.