**Лабораторная работа по теме**

«Знакомство с технологией Windows Forms»

1. Проработайте теоретический материал и ответьте на следующие вопросы:
   1. Какое основное назначение технологии Windows Forms, ее особенности, преимущества и недостатки?
   2. Зачем нужен класс Form? Назовите основные методы, свойства и события данного класса.
   3. Для каких целей используется ключевое слово partial?
   4. Зачем нужен класс STAThreadAttribute?
   5. Для чего в языке используются делегаты? Использование функций обратного вызова (callback). Ковариантность и контравариантность.
   6. Объясните схему работы цепочек делегатов.
   7. Зачем нужны обобщенные делегаты? Основные особенности Action- и Func-делегатов.
   8. Зачем придумали такую конструкцию языка, как анонимные методы? Привидите примеры эффективного использования анонимных методов.
   9. Как используются события в .net? Объясните механизм подписки и отмены подписки на события. Ключевое слово event.
2. Проанализируйте исходный код примера.
3. Создайте приложение по варианту. Используйте различные элементы управления – кнопки, тестовые поля, метки и т.п. Начните с разработки класса Калькулятор. При реализации программного средства используйте делегаты и подписки на события. Не забывайте присваивать корректные имена создаваемым элементам и функциям-обработчикам.
4. Создайте Windows Forms приложение на основе лабораторной №4. Форма должна содержать кнопку генерации коллекции объектов, заданного размера, окно для вывода коллекции, две кнопки для сортировки (убыв, возраст), кнопки для выполнения запросов к колекции и окно вывода их результатов. Для сортировки должен быть один метод и делегат Comparator, который определяет порядок сортировки.
5. Используйте блоки try-catch-finally для проверки корректности вводимых данных в разрабатываемых приложениях.

|  |  |
| --- | --- |
| **Вариант** | **Задание** |
| 1,6,11 | Приложение «Калькулятор для целых». Сложение, вычитание, деление, умножение, двух целых чисел, возведение в степень, хранение значения в памяти. |
| 2, 7, 12 | Приложение «Калькулятор для вещественных». Сложение, вычитание, деление, умножение двух вещественных чисел, извлечение корня, sin, cos, сохранение значения в памяти. |
| 3, 8, 13 | Приложение «Калькулятор для текста». Сложение, вычитание, деление, умножение двух строк. |
| 4, 9, 14 | Приложение «Бинарный калькулятор». Логические операции И, ИЛИ, XOR (исключающее ИЛИ), НЕ для двух целых чисел. Представление чисел в разных системах счисления (двоичная, десятичная, шестнадцатеричная). |
| 5, 10, 15 | Приложение «Процентный калькулятор». Пользователь может ввести Число, Точность, Процент. Операции: процент от числа, прибавить процент к числу, вычесть процент от числа, найти сколько процентов от числа 1 составляет число 2. |

Пример кода:

|  |
| --- |
| using System;  using System.Collections.Generic;  namespace Lab5  {  public class Test  {  public static void Main()  {  try  {  Calculator calc = new Calculator();  calc.AddOperation("\*", (a, b) => a \* b);  calc.AddOperation("/", (a, b) => a / b);  Console.WriteLine(calc.Calculate("\*", 5, 10));  Console.WriteLine(calc.Calculate("/", 5, 10));  Console.WriteLine(calc.Calculate("+", 5, 10));  Console.WriteLine(calc.Calculate("-", 5, 10));  Console.WriteLine(calc.Calculate("^", 5, 10));  calc.RemoveOperation("-");  }  catch (Exception ex)  {  Console.WriteLine(ex.Message);  }  }  }  public class Calculator  {  #region Fields  private readonly Dictionary<string, Func<int, int, int>> \_operations;  #endregion  #region Constructors  public Calculator()  {  \_operations = new Dictionary<string, Func<int, int, int>>  {  {"+", (a, b) => a + b},  {"-", (a, b) => a - b}  };  }  #endregion  #region Methods  public void AddOperation(string operation, Func<int, int, int> body)  {  if (\_operations.ContainsKey(operation))  throw new ArgumentException($"Operation {operation} already exists", "operation");  \_operations.Add(operation, body);  }  public int Calculate(string operation, int a, int b)  {  if (!\_operations.ContainsKey(operation))  throw new ArgumentException($"Operation {operation} is not defined", "operation");  return \_operations[operation](a, b);  }  public IEnumerable<string> GetOperations()  {  return \_operations.Keys;  }  public bool IsOperationExist(string operation)  {  return \_operations.ContainsKey(operation);  }  public void RemoveOperation(string operation)  {  if (!\_operations.ContainsKey(operation))  throw new ArgumentException($"Operation {operation} is not defined", "operation");  \_operations.Remove(operation);  }  #endregion  }  } |