Дисциплина «Программирование корпоративных систем» Рабочая тетрадь 6

Делегаты, лямбда-выражения, события

|  |  |
| --- | --- |
| Ознакомьтесь с теоретическим материалом лекции №6 | |
| **Задание 1** | |
| ***Задача:*** | |
|  | **Использование делегатов для сортировки массива**  Напишите программу, которая сортирует массив строк по разным критериям с использованием делегатов. Программа должна предложить пользователю выбрать один из следующих способов сортировки:   * По длине строки (от самой короткой до самой длинной). * По алфавиту. * По количеству гласных букв в строке.   Создайте соответствующий делегат и передайте его в метод сортировки.  Сопроводите решение текстовым объяснением использования вами механизма делегатов. |
| ***Решение:*** | |
|  | using System;  using System.Linq;  public class Program  {  public delegate string[] SortDelegate(string[] array);  public static void Main()  {  string[] array = { "apple", "banana", "cherry", "date", "elderberry", "fig", "grape" };  while (true)  {  Console.WriteLine("\nВыберите способ сортировки:");  Console.WriteLine("1. По длине строки");  Console.WriteLine("2. По алфавиту");  Console.WriteLine("3. По количеству гласных букв");  Console.WriteLine("4. Выход");  int choice;  if (!int.TryParse(Console.ReadLine(), out choice) || choice < 1 || choice > 4)  {  Console.WriteLine("Неверный выбор. Попробуйте снова.");  continue;  }  if (choice == 4) break;  SortDelegate sorter = choice switch  {  1 => SortByLength,  2 => SortAlphabetically,  3 => SortByVowelCount,  \_ => null  };  if (sorter != null)  {  string[] sortedArray = sorter(array);  Console.WriteLine("Отсортированный массив:");  foreach (var str in sortedArray)  {  Console.WriteLine(str);  }  }  }  }  public static string[] SortByLength(string[] array)  {  return array.OrderBy(s => s.Length).ToArray();  }  public static string[] SortAlphabetically(string[] array)  {  return array.OrderBy(s => s).ToArray();  }  public static string[] SortByVowelCount(string[] array)  {  return array.OrderBy(s => s.Count(c => "aeiouAEIOU".Contains(c))).ToArray();  }  }  Данная программа на языке C# демонстрирует использование делегатов для сортировки массива строк различными способами. Программа предлагает пользователю выбрать метод сортировки из трех предложенных вариантов: по длине строки, по алфавиту или по количеству гласных букв в строке. Делегат SortDelegate объявлен для представления метода, который принимает массив строк и возвращает отсортированный массив строк. Отображается меню с вариантами сортировки, и ввод пользователя проверяется на корректность. Если ввод неверный, программа повторяет запрос. В зависимости от выбора пользователя, устанавливается соответствующий метод сортировки. |
| ***Ответ:*** | |
|  |  |
| **Задание 2** | |
| ***Задача:*** | |
|  | **Фильтрация строк с использованием лямбда-выражений**  Напишите программу, которая фильтрует строки в массиве по указанным пользователем критериям с использованием лямбда-выражений. Программа должна предложить пользователю выбрать один из следующих вариантов фильтрации:   * Строки, длина которых больше указанного числа. * Строки, начинающиеся с определенной буквы. * Строки, содержащие заданное количество гласных букв или больше.   Реализуйте фильтрацию с помощью лямбда-выражений.  Сопроводите решение текстовым объяснением использования вами механизма лямбда-выражений. |
| ***Решение:*** | |
|  | using System;  using System.Linq;  public class Program  {  public static void Main()  {  string[] array = { "apple", "banana", "cherry", "date", "elderberry", "fig", "grape" };  while (true)  {  Console.WriteLine("\nВыберите критерий фильтрации:");  Console.WriteLine("1. Длина строки больше указанного числа");  Console.WriteLine("2. Строки, начинающиеся с определенной буквы");  Console.WriteLine("3. Строки, содержащие заданное количество гласных букв или больше");  Console.WriteLine("4. Выход");  int choice;  if (!int.TryParse(Console.ReadLine(), out choice) || choice < 1 || choice > 4)  {  Console.WriteLine("Неверный выбор. Попробуйте снова.");  continue;  }  if (choice == 4) break;  switch (choice)  {  case 1:  Console.WriteLine("Введите минимальную длину строки:");  int minLength = int.Parse(Console.ReadLine());  var filteredByLength = array.Where(s => s.Length > minLength).ToArray();  PrintArray(filteredByLength);  break;  case 2:  Console.WriteLine("Введите начальную букву:");  char startChar = char.Parse(Console.ReadLine());  var filteredByStartChar = array.Where(s => s.StartsWith(startChar)).ToArray();  PrintArray(filteredByStartChar);  break;  case 3:  Console.WriteLine("Введите минимальное количество гласных букв:");  int minVowelCount = int.Parse(Console.ReadLine());  var filteredByVowelCount = array.Where(s => s.Count(c => "aeiouAEIOU".Contains(c)) >= minVowelCount).ToArray();  PrintArray(filteredByVowelCount);  break;  default:  Console.WriteLine("Неверный выбор.");  break;  }  }  }  private static void PrintArray(string[] array)  {  Console.WriteLine("Отфильтрованный массив:");  foreach (var str in array)  {  Console.WriteLine(str);  }  }  }  Программа использует лямбда-выражения для фильтрации строк по заданным критериям. Пользователь может выбирать критерии фильтрации несколько раз, пока не выберет выход. Лямбда-выражение s => s.Length > minLength определяет условие фильтрации, где s представляет каждую строку в массиве, и фильтрация выполняется для строк, длина которых больше заданного minLength.  Лямбда-выражение s => s.StartsWith(startChar) определяет условие фильтрации, где s представляет каждую строку в массиве, и фильтрация выполняется для строк, начинающихся с заданного символа startChar.  Лямбда-выражение s => s.Count(c => "aeiouAEIOU".Contains(c)) >= minVowelCount определяет условие фильтрации, где s представляет каждую строку в массиве, и фильтрация выполняется для строк, содержащих заданное количество гласных букв или больше. |
| ***Ответ:*** | |
|  |  |

|  |  |
| --- | --- |
| **Задание 3** | |
| ***Задача:*** | |
|  | **Напишите программу, которая моделирует работу термостата.**  Устройство отслеживает изменение температуры и генерирует события, если:   * Температура превышает заданное максимальное значение. * Температура падает ниже заданного минимального значения.   Программа должна:   * Позволить пользователю задать минимальную и максимальную температуры. * Реализовать два обработчика событий: один для предупреждения о перегреве, другой — о переохлаждении. * Симулировать изменение температуры в случайных пределах.   Сопроводите решение текстовым объяснением использования вами механизма событий. |
| ***Решение:*** | |
|  | using System;  using System.Threading;  namespace ThermostatSimulation  {  public class Thermostat  {  public event EventHandler<TemperatureEventArgs> TemperatureChanged;  public double MinTemperature { get; private set; }  public double MaxTemperature { get; private set; }  public double CurrentTemperature { get; private set; }  private Random \_random = new Random();  public Thermostat(double minTemp, double maxTemp)  {  MinTemperature = minTemp;  MaxTemperature = maxTemp;  }  public void UpdateTemperature()  {  CurrentTemperature = SimulateTemperature();  OnTemperatureChanged(new TemperatureEventArgs(CurrentTemperature));  }  private double SimulateTemperature()  {  return Math.Round(  \_random.NextDouble() \* (MaxTemperature + 5 - (MinTemperature - 5)) + (MinTemperature - 5),  2  );  }  protected virtual void OnTemperatureChanged(TemperatureEventArgs e)  {  TemperatureChanged?.Invoke(this, e);  }  }  public class TemperatureEventArgs : EventArgs  {  public double Temperature { get; private set; }  public TemperatureEventArgs(double temperature)  {  Temperature = temperature;  }  }  class Program  {  static void Main(string[] args)  {  Console.Write("Введите минимальную температуру: ");  double minTemp = Convert.ToDouble(Console.ReadLine());  Console.Write("Введите максимальную температуру: ");  double maxTemp = Convert.ToDouble(Console.ReadLine());  Thermostat thermostat = new Thermostat(minTemp, maxTemp);  thermostat.TemperatureChanged += (sender, e) =>  {  Console.WriteLine($"Текущая температура: {e.Temperature:F2}°C");  if (e.Temperature > maxTemp)  {  Console.ForegroundColor = ConsoleColor.Red;  Console.WriteLine($"ВНИМАНИЕ: Перегрев!");  Console.ResetColor();  }  else if (e.Temperature < minTemp)  {  Console.ForegroundColor = ConsoleColor.Blue;  Console.WriteLine($"ВНИМАНИЕ: Переохлаждение!");  Console.ResetColor();  }  };  for (int i = 0; i < 10; i++)  {  thermostat.UpdateTemperature();  Thread.Sleep(1000);  }  }  }  }  Пользователь задает температурные границы, после чего симулятор генерирует случайные значения температуры и отслеживает выход за допустимые пределы. При достижении критических значений выводятся цветовые предупреждения.  Делегат TemperatureEventHandler объявлен для представления метода, который будет обрабатывать события, связанные с температурой. Этот делегат принимает два параметра: object sender, представляющий источник события, и TemperatureEventArgs e, содержащий данные о событии.  Два события объявлены с использованием делегата TemperatureEventHandler:  Overheat - событие, которое срабатывает, когда температура превышает максимальное значение.  Underheat - событие, которое срабатывает, когда температура падает ниже минимального значения |
| ***Ответ:*** | |

Изображение выглядит как текст, снимок экрана, Шрифт

Автоматически созданное описание