Ильяхова Алиса, Б9122-02.03.01сцт

Лабораторная работа №1. Применение делегатов

Вариант 9

1. Цель и задачи работы

Цель:

Изучить возможности применения делегатов в языке C#.

Задачи:

- Освоить принципы работы с делегатами;
- Освоить основные направления применения делегатов;
- Изучить способы использования делегатов совместно с потоками.

2. Реализация индивидуального задания

Согласно таблице индивидуальных заданий (стр. 12–13), для **варианта 9** требуется реализовать делегат следующей сигнатуры:

Action<Func<int>, char, char>

Это означает:

- метод **не возвращает значение** (Action);
- принимает три параметра:
    - Func<int> — делегат без параметров, возвращающий int;
    - два символа типа char.

2.1. Объявление пользовательского делегата

```
public delegate void MyDelegate9(Func<int> intProvider, char c1, char c2);
```

2.2. Реализация методов, соответствующих сигнатуре делегата

Были реализованы три метода:

Метод 1: Шифрование символов

```
static void EncryptChars(Func<int> intProvider, char first, char second)
    {
        int shift = intProvider(); // Получаем сдвиг
        char encrypted1 = (char)(first + shift);
        char encrypted2 = (char)(second + shift);
        Console.WriteLine($"[Шифрование] Исходные: '{first}', '{second}' => Зашифровано:
'{encrypted1}', '{encrypted2}'");
    }
```

Метод 2: Сравнение символов с порогом

```
static void CompareWithThreshold(Func<int> intProvider, char a, char b)
    {
        int threshold = intProvider();
        int diff = Math.Abs(a - b);
        Console.WriteLine($"[Сравнение] Символы: '{a}' ({(int)a}), '{b}' ({(int)b}) =>
Разница: {diff}");
        if (diff > threshold)
            Console.WriteLine($"Разница ({diff}) > порога ({threshold}) => Символы сильно
отличаются.");
        else
            Console.WriteLine($"Разница ({diff}) ≤ порога ({threshold}) => Символы
близки.");
    }
```

Метод 3: Анализ ASCII-кодов

```
static void AnalyzeAsciiCodes(Func<int> intProvider, char x, char y)
    {
        int rand = intProvider();
        int sum = x + y + rand;
        Console.WriteLine($"[Анализ] '{x}' + '{y}' + {rand} = {sum} (ASCII: {x}={x},
{y}={y})");
    }
```

2.3. Использование делегата с потоками

Для демонстрации взаимодействия с потоками реализован метод ThreadedHandler:

```
static void ThreadedHandler(Func<int> provider, char a, char b)
    {
        Console.WriteLine($"Поток #{Thread.CurrentThread.ManagedThreadId} начал работу.");
        Thread.Sleep(500); // имитация работы
        EncryptChars(provider, a, b);
        CompareWithThreshold(provider, a, b);
```

```
        Console.WriteLine($"Поток #{Thread.CurrentThread.ManagedThreadId} завершил
работу.");
    }
```

2.4. Основной метод Main

```
static void Main(string[] args)
    {
        // Источник целого числа: случайное число от 1 до 10
        Func<int> randomInt = () => new Random().Next(1, 11);

        // Создаём экземпляры делегата
        MyDelegate9 del1 = EncryptChars;
        MyDelegate9 del2 = CompareWithThreshold;
        MyDelegate9 del3 = AnalyzeAsciiCodes;

        Console.WriteLine("=== Вызов методов через делегат ===");
        del1(randomInt, 'A', 'Z');
        del2(() => 5, 'X', 'M');
        del3(() => 100, 'a', 'b');

        Console.WriteLine("\n=== Демонстрация использования с потоками ===");

        // Создаём и запускаем поток с использованием делегата
        Thread thread = new Thread(() =>
        {
            MyDelegate9 threadDel = ThreadedHandler;
            threadDel(() => 3, 'H', 'i');
        });

        thread.Start();
        thread.Join(); // дожидаемся завершения

        Console.WriteLine("\n=== Готово ===");
    }
```

3. Ответы на контрольные вопросы

1. **Что такое тип делегата? Какой аналог типа делегата существует в C++?**

Делегат — это тип, представляющий ссылку на метод с определённой сигнатурой. Он позволяет передавать методы как параметры. Аналогом в C++ являются **указатели на функции**.

2. **Опишите основные направления использования делегатов.**
    a. Обработка событий (например, в GUI-приложениях);
    b. Реализация обратных вызовов (callback);

c. Передача поведения как параметра (например, в алгоритмах сортировки);
d. Запуск методов в отдельных потоках или задачах.

3. **Какие механизмы технологии Windows Forms реализованы с использованием делегатов?**

Все **события** (например, Click, TextChanged, Load) в Windows Forms реализованы через делегаты, чаще всего через EventHandler.

4. **Для чего предназначен тип Action<T>? Чем он отличается от Func<T>?**
   a. Action<T> — делегат, **не возвращающий значение** (void);
   b. Func<T> — делегат, **возвращающий значение** (последний обобщённый параметр — тип возврата).
5. **Чем пользовательские делегаты отличаются от библиотечных?**

Пользовательские делегаты объявляются явно под конкретную сигнатуру (delegate void MyDel(...)), тогда как Action и Func — универсальные обобщённые делегаты из стандартной библиотеки. Они **взаимозаменяемы**, если сигнатуры совпадают.

4. Результаты выполнения (консольный вывод)

Пример выполнения программы:

```
=== Вызов методов через делегат ===
[Шифрование] Исходные: 'A', 'Z' => Зашифровано: 'J', 'c'
[Сравнение] Символы: 'X' (88), 'M' (77) => Разница: 11
Разница (11) > порога (5) => Символы сильно отличаются.
[Анализ] 'a' + 'b' + 100 = 295 (ASCII: a=a, b=b)

=== Демонстрация использования с потоками ===
Поток #9 начал работу.
[Шифрование] Исходные: 'H', 'i' => Зашифровано: 'K', 'l'
[Сравнение] Символы: 'H' (72), 'i' (105) => Разница: 33
Разница (33) > порога (3) => Символы сильно отличаются.
Поток #9 завершил работу.

=== Готово ===
```