МИНИCTEPCTBO НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное автономное образовательное учреждение

высшего образования

«СЕВЕРО-КАВКАЗСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

Институт перспективной инженерии

Департамент цифровых, робототехнических систем и электроники

Отчет к практической работе № 13

Тема: **«Применение делегатов»**

Дисциплина: **«Алгоритмы и структуры данных»**

|  |  |  |
| --- | --- | --- |
|  |  | **Выполнил:** |
|  |  | Студент группы ПИЖ-б-о-23-1, направление подготовки: 09.03.04 «Программная инженерия»  Панчешный Александр Алексеевич  **Проверил:**  Доцент ДЦРСИЭ  Николаев Евгений Иванович |

Ставрополь 2025

**ЛАБОРАТОРНАЯ РАБОТА 13. ПРИМЕНЕНИЕ ДЕЛЕГАТОВ В C#**

**Цель работы:**  
Овладеть концепцией делегатов в языке C# и научиться применять их для решения практических задач.

**Задачи лабораторной работы:**

* освоить принципы работы с делегатами;
* освоить основные направления применения делегатов;
* изучить способы использования делегатов совместно с потоками.

**Ход работы**

**Индивидуальное задание**:

using System;

using System.Collections.Generic;

class Program

{

*// Пользовательский делегат с требуемой сигнатурой*

public delegate bool CustomDelegate(Action<List<int>> action);

static void Main()

{

*// Создаем экземпляр делегата и связываем его с методом ProcessNumbers*

CustomDelegate processor = ProcessNumbers;

*// Метод Action для обработки списка чисел*

Action<List<int>> printAction = list =>

{

Console.WriteLine("Элементы списка:");

foreach (var num in list)

{

Console.Write(num + " ");

}

Console.WriteLine();

};

*// Вызов делегата*

bool result = processor(printAction);

Console.WriteLine($"Результат обработки: {result}");

*// Второй пример с другим Action*

Action<List<int>> sumAction = list =>

{

int sum = 0;

foreach (var num in list)

{

sum += num;

}

Console.WriteLine($"Сумма элементов: {sum}");

};

result = processor(sumAction);

Console.WriteLine($"Результат обработки: {result}");

}

*// Метод, соответствующий сигнатуре делегата*

static bool ProcessNumbers(Action<List<int>> action)

{

try

{

*// Создаем список чисел*

List<int> numbers = new List<int> { 1, 2, 3, 4, 5 };

*// Вызываем переданный Action*

action(numbers);

*// Возвращаем true, если обработка прошла успешно*

return true;

}

catch

{

*// В случае ошибки возвращаем false*

return false;

}

}

}

**Контрольные вопросы:**

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

**Делегат** в C# — это тип, представляющий ссылку на метод с определенной сигнатурой (списком параметров и возвращаемым значением). Он аналогичен **указателям на функции** в C++, но с ключевыми отличиями:

* Безопасность типов (проверка на этапе компиляции).
* Поддержка множественных методов (через += и -=).
* Интеграция с событиями и LINQ.

**2. Основные направления использования делегатов**

1. Обратные вызовы (Callbacks)
   * Передача методов как параметров (например, в асинхронных операциях).
2. Обработка событий
   * Основа событийной модели в .NET (Button.Click, Timer.Elapsed).
3. Гибкость архитектуры
   * Стратегии, подписки, плагины (например, сортировка с разными критериями).
4. LINQ и функциональное программирование
   * Использование Func<T> и Predicate<T> для фильтрации коллекций.
5. Многопоточность
   * Передача задач в потоки через ThreadStart или Task.

**3. Механизмы Windows Forms, использующие делегаты**

**Обработчики событий** (Button.Click, TextBox.TextChanged):

button1.Click += (sender, e) => MessageBox.Show("Клик!");

**Асинхронное обновление UI** через Control.Invoke:

this.Invoke((Action)(() => label1.Text = "Обновлено!"));

**4. Тип**Action<T>**vs**Func<T>

|  |  |  |
| --- | --- | --- |
| **Критерий** | Action<T> | Func<T> |
| |  |  | | --- | --- | | **Возврат значения** |  | | Не возвращает (void) | Возвращает значение |
| **Пример** | Action<int> — void M(int x) | Func<int, bool> — bool M(int x) |
| **Использование** | Для действий | Для вычислений |

**Пример:**

Action<string> log = msg => Console.WriteLine(msg); *// Без возврата*

Func<int, int, int> sum = (a, b) => a + b; *// Возвращает int*