МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное автономное образовательное учреждение высшего образования

«Дальневосточный федеральный университет»
(ДВФУ)

ИНСТИТУТ МАТЕМАТИКИ И КОМПЬЮТЕРНЫХ ТЕХНОЛОГИЙ

Департамент математического и компьютерного моделирования

Лабораторная работа №1

Вариант IV по дисциплине
«Суперкомпьютеры и параллельная обработка данных»

Направление подготовки
02.03.01 «Математика и компьютерные науки»

Выполнила студентка
группы Б9122-02.03.01сцт
Винницкая Д.С.
(ФИО) *(подпись)*

« 30 » сентября 20 25 г.

г. Владивосток
2025

1 Цель и задачи лабораторной работы

Цель: изучить возможности применения делегатов в языке C#.

Задачи:

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

2 Реализация индивидуального задания

Согласно варианту задания, требуется реализовать **пользовательский тип делегата** со следующей сигнатурой:

Func<Action<float>, int, float, bool>

Это означает, что делегат должен принимать три аргумента:

1. Action<float> — метод, принимающий float и ничего не возвращающий;
2. int — целое число;
3. float — дробное число;

и возвращать значение типа bool. Для соответствия требованию «пользовательский тип делегата» был объявлен собственный делегат MyDelegate, а не использован встроенный Func<...>.

2.1 Листинг программного кода

Ниже приведён полный исходный код программы.

```
using System;

public delegate bool MyDelegate(Action<float> processor, int count,
    float value);

class Program
{

    static bool ProcessWithLogging(Action<float> action, int count,
        float value)
    {
        Console.WriteLine($Метод"[ 1] Запуск: count={count},
            value={value}");
        for (int i = 0; i < count; i++)
        {
            action(value + i * 0.5f);
        }
        return value > 0 && count > 0;
    }


    static bool ProcessIfValid(Action<float> action, int count, float
        value)
    {
        if (count >= 2 && value >= 5.0f)
        {
            Console.WriteLine($Метод"[ 2] Условиевыполнено .
                Выполняемдействие ...");
            action(value * count);
            return true;
        }
        Console.WriteLine($Метод"[ 2] УсловиеНЕвыполнено .");
        return false;
    }

    static void PrintValue(float x)
    {
        Console.WriteLine($" Обработказначения : {x:F2}");
    }

    static void Main(string[] args)
    {
        Console.WriteLineЛабораторная(" работа№ 1: Делегатыв C# \n");

        MyDelegate del1 = ProcessWithLogging;
        MyDelegate del2 = ProcessIfValid;

        Action<float> printer = PrintValue;

        Console.WriteLine→(" Вызовметода 1:");
        bool res1 = del1(printer, 3, 2.0f);
```

```
47                  Console.WriteLine($"Результат": {res1}\n");
48
49                  Console.WriteLine(" Вызовметода  2 неудачный():");
50                  bool res2 = del2(printer, 1, 3.0f);
51                  Console.WriteLine($"Результат": {res2}\n");
52
53                  Console.WriteLine(" Вызовметода  2 успешный():");
54                  bool res3 = del2(printer, 4, 6.0f);
55                  Console.WriteLine($"Результат": {res3}\n");
56
57                  MyDelegate lambdaDel = (act, c, v) =>
58                  {
59                      Console.WriteLine($"Лямбда"[] c={c}, v={v}");
60                      act(v + c);
61                      return v + c > 10;
62                  };
63
64                  Console.WriteLine(" Вызовчерезлямбду  :");
65                  bool res4 = lambdaDel(printer, 5, 6.5f);
66                  Console.WriteLine($"Результат" лямбды: {res4}");
67              }
68          }
```

Листинг 1: Реализация пользовательского делегата и его использование

2.2 Описание

Объявление делегата выглядит следующим образом:

```
1   public delegate bool MyDelegate(Action<float> processor, int count, float
        value);
```

Данный тип описывает метод, который:

принимает делегат Action<float> (метод, принимающий float и не возвращающий значение);

- принимает целочисленный параметр int count;
- принимает дробное число float value;
- возвращает логическое значение bool.

Были реализованы два метода, полностью соответствующих данной сигнатуре:

1. ProcessWithLogging — выполняет циклический вызов переданного Action<flo count раз с изменяющимся значением (value + i * 0.5f). Возвращает true, если оба входных параметра положительны.

2. ProcessIfValid — реализует условную логику: действие выполняется только при выполнении условия count >= 2 и value >= 5.0f. В противном случае метод возвращает false.

В качестве реализации Action<float> используется метод PrintValue, который выводит переданное значение на консоль с форматированием до двух знаков после запятой.

В методе Main:

1. создаются экземпляры делегата del1 и del2, ссылающиеся на реализованные методы;
2. переменная printer хранит ссылку на метод PrintValue;
3. демонстрируются вызовы делегатов с различными наборами аргументов, включая успешный и неуспешный сценарии;
4. дополнительно показано использование лямбда-выражения, реализующего ту же сигнатуру, что и MyDelegate.

2.3 Результат выполнения программы

При запуске программы в консоли получен следующий вывод:

```
→Вызовметода

    1:Метод
[ 1] Запуск: count=3, value=2Обработказначения
     : 2,00Обработказначения
     : 2,50Обработказначения
     : 3,00Результат
: True→Вызовметода

    2 неудачный():Метод
[ 2] УсловиеНЕвыполнено .Результат
: False→Вызовметода

    2 успешный():Метод
[ 2] Условиевыполнено . Выполняемдействие ...Обработказначения
     : 24,00Результат
: True→Вызовчерезлямбду

    :Лямбда
[] c=5, v=6,5Обработказначения
     : 11,50Результатлямбды
 : True
```

Листинг 2: Консольный вывод программы

1. **Вызов метода** ProcessWithLogging:
   Делегат del1 был вызван с параметрами: printer (ссылка на PrintValue), count = 3, value = 2.0.
   Внутри метода запустился цикл, который трижды вызвал переданный Action<float>, передавая значения 2.0, 2.5 и 3.0 (с шагом 0.5).
   Все значения были успешно выведены, а метод вернул True, так как оба входных параметра (count и value) положительны. Это подтверждает корректную работу делегата при передаче и многократном вызове метода-обработчика.

2. **Неуспешный вызов метода** ProcessIfValid:
   Был передан набор аргументов: count = 1, value = 3.0.
   Условие метода (count >= 2 value >= 5.0f) не выполнилось, поэтому действие action(...) не было вызвано, и метод вернул False.
   Это демонстрирует способность делегата инкапсулировать логику с

условным выполнением, что особенно полезно при реализации стратегий или валидации.

3. **Успешный вызов метода** ProcessIfValid:
   Параметры: $\text{count} = 4$, $\text{value} = 6.0$.
   Условие выполнено, поэтому внутри метода был вызван $\text{action}(\text{value} * \text{count}) \rightarrow \text{action}(24.0)$.
   Значение 24.00 было выведено, а метод вернул True.
   Этот сценарий показывает, как делегат может не только передавать поведение, но и комбинировать его с внутренней логикой, включая арифметические операции и принятие решений.

4. **Вызов через лямбда-выражение**:
   Был создан анонимный делегат с помощью лямбды: $(\text{act}, \text{c}, \text{v}) => \{ \ldots \}$.
   При вызове с аргументами $\text{c} = 5$, $\text{v} = 6.5$ лямбда вывела информацию о параметрах, вызвала $\text{act}(\text{v} + \text{c}) \rightarrow \text{act}(11.5)$, и вернула True, так как $11.5 > 10$.

3 Контрольные вопросы

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

   Делегат в C# — это тип, который хранит ссылку на метод с определённой сигнатурой. Он позволяет передавать методы как параметры, вызывать их позже или даже объединять несколько методов в цепочку. В C++ ближайший аналог — это std::function или указатель на функцию.

2. **Опишите основные направления использования делегатов.**

   Делегаты используются для обратных вызовов (callbacks), обработки событий (например, кликов в интерфейсе), реализации паттернов вроде Strategy или Observer, а также в LINQ и асинхронном программировании. Они делают код гибким и расширяемым.

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

   В Windows Forms вся система событий построена на делегатах. Например, событие Click у кнопки — это делегат типа EventHandler. При подписке через button.Click += MyHandler вы добавляете метод в список вызовов делегата.

4. **Для чего предназначен тип** Action<T>**? Чем он отличается от** Func<T>**?**

   Action<T> — это готовый делегат для методов, которые ничего не возвращают (void). Func<T> — для методов, которые возвращают значение. Последний тип в Func всегда — тип возвращаемого результата.

5. **Чем пользовательские делегаты отличаются от библиотечных?**

   Пользовательские делегаты объявляются вручную с помощью ключевого слова delegate и имеют понятное имя (например, DataProcessor). Библиотечные (Action, Func) — универсальные и подходят для большинства случаев, но могут быть менее читаемыми в сложной логике. По сути, они делают одно и то же — просто разный стиль.