# ЛАБ8

1. Делегат — это тип, который представляет ссылки на методы [с](https://learn.microsoft.com/en-us/dotnet/csharp/language-reference/builtin-types/reference-types) определенным списком параметров и типом возвращаемого значения. delegate void MyDelegate(int x); Обработчики событий — это не что иное, как методы, вызываемые через делегатов.
2. Делегаты используются для передачи методов в качестве аргументов другим методам. Делегаты обеспечивают гибкость и возможность обратного вызова (callback) в приложениях. Полезно в ситуациях, когда требуется выполнить определенные действия, но точное действие зависит от контекста или условий выполнения.
3. Есть несколько способов присваивания делегату адреса метода:

|  |
| --- |
| С использованием имени метода:  MyDelegate delegateInstance = SomeMethod; |
| С использованием ключевого слова "delegate":  MyDelegate delegateInstance = new MyDelegate(SomeMethod); |
| С использованием лямбда-выражения:  MyDelegate delegateInstance = x => Console.WriteLine(x); |

1. Метод Invoke используется для вызова метода, на который ссылается делегат. Invoke также позволяет передавать параметры методу, на который ссылается делегат.
2. Групповая адресация делегата представляет возможность добавления нескольких методов в делегат, чтобы они вызывались последовательно при вызове делегата. Таким образом, делегат может ссылаться на несколько методов, и все эти методы будут вызваны при вызове делегата.

Для добавления метода в группу делегата используется оператор "+=". При вызове делегата "delegateInstance" будут вызваны все методы SomeMethod1, SomeMethod2 и SomeMethod3 в порядке их добавления.

|  |
| --- |
| MyDelegate delegateInstance = SomeMethod1;  delegateInstance += SomeMethod2;  delegateInstance += SomeMethod3; |

1. Чтобы создать событие в C#, нужно использовать ключевое слово "event". Событие представляет собой механизм, позволяющий классу уведомлять другие классы о возникновении определенного события.
2. Событие представляет собой обертку над делегатом.

Взаимодействие между событиями и делегатами происходит следующим образом:

* Определение делегата: Сначала необходимо определить делегат, который будет использоваться для представления события. Делегат определяет сигнатуру метода, на который может ссылаться и который будет вызываться при возникновении события.
* Определение события: Событие объявляется с использованием ключевого слова event и связывается с делегатом. Это создает связь между событием и одним или несколькими методами, которые будут вызываться при возникновении события.
* Подписка на событие: Чтобы методы могли быть вызваны при возникновении события, им необходимо подписаться на это событие. Это делается путем добавления методов в список обработчиков события с использованием оператора +=. При возникновении события все методы в списке будут вызваны.
* Генерация события: Когда наступает условие, которое должно вызвать событие, код должен явно вызвать событие. Это делается путем вызова делегата события, который в свою очередь вызывает все зарегистрированные методы-обработчики.

1. Лямбда-выражения представляют упрощенную запись анонимных методов. Лямбда-выражения позволяют создать емкие лаконичные методы, которые могут возвращать некоторое значение и которые можно передать в качестве параметров в другие методы.

|  |
| --- |
| (int x, int y) => x + y |

1. Ковариантность делегатов: если у делегата есть метод, который принимает базовый тип данных, то мы можем присвоить этому делегату метод, который принимает производный тип данных. То есть мы можем использовать более специфичные (производные) типы данных вместо более общих (базовых) типов данных.

Контравариантность делегатов: если у делегата есть метод, который принимает производный тип данных, то мы можем присвоить этому делегату метод, который принимает базовый тип данных. То есть мы можем использовать более общие (базовые) типы данных вместо более специфичных (производных) типов данных.

1. Action - это делегат, который представляет метод без возвращаемого значения. Он может принимать от 0 до 16 параметров (обозначаемых как Action, Action<T>, Action<T1, T2>, и так далее, до Action<T1, T2, ..., T16>). Action используется, когда не требуется возвращаемое значение из метода.

Func - это делегат, который представляет метод с возвращаемым значением. Он также может принимать от 0 до 16 параметров (обозначаемых как Func<TResult>, Func<T, TResult>, Func<T1, T2, TResult>, и так далее, до Func<T1, T2, ..., T16, TResult>). Последний тип параметра (TResult) представляет возвращаемый тип значения.