**Вопросы к лабораторной работе №8**

1. Что такое делегат? Как определить делегат?

**Делегат** – это объект, предназначенный для хранения ссылок на методы (указатель на функцию С++).

Тип делегата объявляется с помощью ключевого слова **delegate**. Ниже приведена общая форма объявления делегата:

*delegate возвращаемый\_тип имя (список\_параметров);*

где возвращаемый\_тип обозначает тип значения, возвращаемого методами, которые будут вызываться делегатом;

имя — конкретное имя делегата;

список\_параметров — параметры, необходимые для методов, вызываемых делегатом. Как только будет создан экземпляр делегата, он может вызывать и ссылаться на те методы, возвращаемый тип и параметры которых соответствуют указанным в объявлении делегата.

*public delegate void D(int i)  
class Class1  
 private static void Hello(int i) { }  
 static void Main()  
 D del; //создаём переменную делегата  
 del = Hello; //присваиваем переменной адрес метода  
. //del = new D(Hello)  
 del.Invoke(4); //вызываем метод  
 //del(4)*

2. Назначение делегатов.

1) возможность определять вызываемый метод не при компиляции, а динамически во время выполнения программы;

2) обеспечение связи между объектами по типу «источник —наблюдатель»;

3) создание универсальных методов, в которые можно передавать другие методы;

4) поддержка механизма обратных вызовов.

5) используются для поддержки событий

3. Какие есть способы присваивания делегату адреса метода?

Выше переменной делегата напрямую присваивался метод. Есть еще один способ - создание объекта делегата с помощью конструктора, в который передается нужный метод:

*class Program*

*{*

*delegate int Operation(int x, int y);*

*static void Main(string[] args)*

*{*

*Operation del = Add;*

*Operation del2 = new Operation(Add);*

*Console.Read();*

*}*

*private static int Add(int x, int y) { return x + y; }*

*}*

4. Поясните назначение метода **Invoke()**.

Вызов делегата.

Метод **Invoke()**, определенный в классе Parallel, позволяет выполнять один или несколько методов, указываемых в виде его аргументов.

Ниже приведена простейшая форма его объявления:

*public static void Invoke(params Action[] actions)*

Выполняемые методы должны быть совместимы с описанным ранее делегатом Action. Следовательно, каждый метод, передаваемый методу **Invoke()** в качестве аргумента, не должен ни принимать параметров, ни возвращать значение.

5. Что такое групповая адресация делегата?

Делегат может ссылаться на группу методов, которые связаны между собой в цепочку.

**Групповая адресация** – это способ (возможность) создания списка методов, которые вызываются автоматически при обращении к делегату.

Групповая адресация позволяет сформировать списки (цепочки) вызовов. Это есть эффективным, поскольку:

* можно гибко формировать списки методов, которые должны вызываться по одному вызову делегата;
* удобно формировать списки методов, которые выполняют разные виды работ над некоторым общим объектом;
* улучшается структура программного кода;
* удобно обрабатывать события, которые генерируются системой.

Для организации групповой адресации методов с помощью делегата используются 4 операции:

* операции ‘+’ и ‘+=’ – добавляют метод к списку методов;
* операции ‘−’ и ‘−=’ – удаляют метод из списка методов.

6. Как создать событие?

**Событие** – элемент класса, позволяет ему посылать другим объектам уведомления об изменении своего состояния.

События являются членами класса и объявляются с помощью ключевого слова **event**. Чаще всего для этой цели используется следующая форма:

*event делегат\_события имя\_события;*

где *делегат\_события* обозначает имя делегата, используемого для поддержки события, а *имя\_события* — конкретный объект объявляемого события.

7. Как события связаны с делегатами? Опишите и поясните схему взаимодействия.

Связь с делегатом означает, что метод, обрабатывающий данное событие, должен принимать те же параметры и возвращать тот же тип, что и делегат.

События построены на основе делегатов: с помощью делегатов вызываются методы-обработчики событий. Поэтому создание события в классе состоит из следующих частей:

* описание делегата, задающего сигнатуру обработчиков событий;
* описание события;
* описание метода (методов), инициирующих cобытие.

8. Что такое лямбда-выражения? Приведите пример лямбда-выражения с несколькими параметрами.

Упрощённая запись анонимных методов (нет return)

*(x,y) => x + y;  
i => i \* i;*

**Анонимные функции** – безымянный кодовый блок, передаваемый конструктору делегата.

*delegate int Summator(int b);  
static int result = 0;  
Summator someDelegat = delegate (int number)  
{  
 for (i=0; i<=number; i++)  
 result +=i; //захват переменной  
 return result;  
}*

Operation sum = (x, y) => Console.WriteLine($"{x} + {y} = {x + y}");

sum(1, 2);       // 1 + 2 = 3

sum(22, 14);    // 22 + 14 = 36

delegate void Operation(int x, int y);

9. Что такое ковариантность и контравариантность делегатов? Что это дает?

**Ковариантность** позволяет возвращать из метода объекта, тип которого является производным от типа, возвращаемого делегата.

Ковариантность позволяет передать делегату метод, возвращаемый тип которого является производный от возвращаемого типа делегат. То есть если возвращаемый тип делегата Message, то у метод может иметь в качестве возвращаемого типа класс EmailMessage:

**Контравариантность** предполагает возможность использования метода объекта, тип которого является универсальным по отношению к типу представителя.

Понятия ковариантности и контравариантности связаны с использованием в приложении вместо некоторого типа другой тип, который находится ниже в иерархии наследования.

10. Поясните разницу между встроенными делегатам Action и Func.

**Action** – представляет некоторое действие, которое ничего не возвращает, то есть в качестве возвращаемого типа имеет тип void  
**Func** – возвращает результат действия и может принимать параметры.