**Лабораторная работа № 60-61**

**«Делегаты»**

**Цель работы:** получение навыков работы с файловыми потоками в С#.

**Задания для лабораторной работы по вариантам:**

5. Создайте метод, который генерирует последовательность чисел на основе переданного делегата. Делегат должен определять правило генерации следующего числа в последовательности. Реализуйте несколько правил:

* арифметическая прогрессия (например, начальное число 5, шаг 3: 5, 8, 11, 14, ...).
* геометрическая прогрессия (например, начальное число 2, множитель 3: 2, 6, 18, 54, ...).
* последовательность Фибоначчи (каждое следующее число — сумма двух предыдущих: 1, 1, 2, 3, 5, 8, ...).

class Program

{

public delegate int NextNumberDelegate();

static void Main(string[] args)

{

Console.WriteLine("Арифметическая прогрессия:");

GenerateSequence(ArithmeticProgression(5, 3), 10);

Console.WriteLine("\nГеометрическая прогрессия:");

GenerateSequence(GeometricProgression(2, 3), 10);

Console.WriteLine("\nПоследовательность Фибоначчи:");

GenerateSequence(FibonacciSequence(), 10);

}

static void GenerateSequence(NextNumberDelegate nextNumber, int count)

{

for (int i = 0; i < count; i++)

{

Console.Write(nextNumber() + " ");

}

Console.WriteLine();

}

static NextNumberDelegate ArithmeticProgression(int start, int step)

{

int current = start;

return () =>

{

int result = current;

current += step;

return result;

};

}

static NextNumberDelegate GeometricProgression(int start, int multiplier)

{

int current = start;

return () =>

{

int result = current;

current \*= multiplier;

return result;

};

}

static NextNumberDelegate FibonacciSequence()

{

int a = 0, b = 1;

return () =>

{

int result = a;

int temp = a;

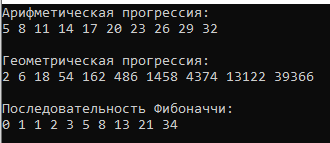
a = b; b = temp + b;

return result;

};

}

}



**Контрольные вопросы:**

1.Что такое делегат в C#?

Делегат — это тип, который представляет ссылку на метод с определенной сигнатурой. Т.е. делегирует обращение к этому методу.

2. Как объявить делегат? Приведите пример.

3. delegate bool FilterDelegate(int number);

Как связать делегат с методом?

MyDelegate del = MyMethod;

4. Какие типы методов могут быть связаны с делегатом?

Делегаты могут ссылаться на:

Статические методы.

Методы экземпляра (методы, принадлежащие конкретному объекту).

Анонимные методы.

Лямбда-выражения.

Методы с параметрами и возвращаемыми значениями.

5. Что такое групповые делегаты (multicast)?

Групповые делегаты — это делегаты, которые могут ссылаться на несколько методов одновременно. Когда вызывается такой делегат, все методы в его списке вызовов выполняются последовательно.

6.Как добавить метод в список вызовов группового делегата?

Метод можно добавить к групповому делегату с помощью оператора +=.

public delegate void MyDelegate();

MyDelegate del = Method1;

del += Method2; // Добавление метода

7. Как удалить метод из списка вызовов группового делегата?

Метод можно удалить из группового делегата с помощью оператора -=.

del -= Method1; // Удаление метода

8. Может ли делегат ссылаться на метод с возвращаемым значением? Если да, приведите пример.

делегат может ссылаться на методы с возвращаемым значением. Например:

public delegate int MyDelegate(int x, int y);

public class Example

{

public int Add(int a, int b)

{

return a + b;

}

}

Example example = new Example();

MyDelegate del = example.Add;

int result = del(5, 10); // result будет равен 15

9. Может ли делегат ссылаться на метод с параметрами? Если да, приведите пример.

делегат может ссылаться на методы с параметрами.

public delegate void MyDelegate(string message);

public class Example

{

public void PrintMessage(string msg)

{

Console.WriteLine(msg);

}

}

Example example = new Example();

MyDelegate del = example.PrintMessage;

del("Hello, World!"); // Вывод: Hello, World!

10. Как передать делегат в качестве параметра метода?

Делегат можно передать как параметр метода, указав его тип в сигнатуре метода.

public void Execute(MyDelegate del)

{

del();

}

// Использование

Execute(del);

11. Что такое анонимные методы и как они связаны с делегатами?

Анонимные методы — это методы без имени, которые можно определить прямо в месте их использования. Они позволяют создавать функциональность "на лету" и могут быть присвоены делегатам.

MyDelegate del = delegate { Console.WriteLine("Это анонимный метод."); };

12. Как делегаты влияют на производительность по сравнению с прямым вызовом?

Делегаты могут быть немного медленнее, чем прямые вызовы методов из-за накладных расходов на создание и вызов делегата. Однако разница часто незначительна и компенсируется гибкостью и возможностями, которые они предоставляют.

13. Какие вы считаете есть ограничение у делегатов?

Делегаты могут ссылаться только на методы с одинаковой сигнатурой.

Нельзя использовать делегаты для передачи состояния (например, переменные не могут быть переданы по ссылке).

Сложность отладки при использовании многократных вызовов.

14. Какие альтернативы делегатам есть в других языках программирования?

В других языках программирования аналогами делегатов являются:

Функциональные указатели в C/C++.

Callback-функции в JavaScript и других языках.

Функции высшего порядка в функциональных языках (например, Haskell).

15. Как делегаты связаны с функциональным программированием?

Делегаты позволяют передавать функции как параметры, что является основным аспектом функционального программирования. Это позволяет создавать более гибкие и модульные программы.

16. В чем разница между делегатом и указателем на функцию в C++?

Делегаты являются типобезопасными и проверяются во время компиляции, тогда как указатели на функции в C++ не имеют такой безопасности.

Делегаты могут ссылаться на методы экземпляров и статические методы, тогда как указатели на функции могут ссылаться только на функции.

17. Как вызвать делегат?

del(); // Вызов делегата

18. Что произойдет, если вызвать делегат, который не ссылается ни на один метод?

произойдёт исключение NullReferenceException.

19. Может ли делегат ссылаться на метод экземпляра класса? Если да, приведите пример.

делегат может ссылаться на метод экземпляра класса.

public class Example

{

public void InstanceMethod()

{

Console.WriteLine("Это метод экземпляра.");

}

}

Example example = new Example();

Action del = example.InstanceMethod; // Ссылка на метод экземпляра

del(); // Вызов метода

20. Может ли делегат ссылаться на статический метод? Если да, приведите пример.

делегат может ссылаться на статические методы.

public class Example

{

public static void StaticMethod()

{

Console.WriteLine("Это статический метод.");

}

}

Action del = Example.StaticMethod; // Ссылка на статический метод

del(); // Вызов метода

21. Как создать делегат, который принимает несколько параметров?

public delegate void MyDelegate(int x, int y);

22. Как создать делегат, который возвращает значение?

public delegate int MyDelegate(int x, int y);

23. Что такое обобщенные делегаты (Func и Action)?

Action — это обобщённый делегат, который представляет метод, не возвращающий значения и принимающий параметры.

Func — это обобщённый делегат, который представляет метод, возвращающий значение и может принимать параметры.

24. Как использовать делегат Func в C#?

Func<int, int, int> add = (x, y) => x + y;

int result = add(5, 10); // result будет равен 15

25. Как использовать делегат Action в C#?

Action<string> print = message => Console.WriteLine(message);

print("Hello, World!"); // Вывод: Hello, World!

26. Как делегаты используются в WPF и WinForms для обработки событий?

В WPF и WinForms события реализованы через делегаты. Когда событие происходит (например, кнопка нажата), вызывается соответствующий обработчик событий — метод, связанный с событием через делегат.

27. Как создать кастомный делегат с переменным числом параметров? (params)

Вы можете использовать ключевое слово params для создания метода с переменным числом параметров:

public delegate void MyDelegate(params int[] numbers);

28. Какие преимущества предоставляют делегаты по сравнению с прямым вызовом методов?

• Гибкость: Позволяют динамически изменять поведение программы

• Модульность: Упрощают создание модульных приложений с возможностью обработки событий.

• Поддержка многократных вызовов: Позволяют вызывать несколько методов одним вызовом.

• Обработка событий: Упрощают реализацию паттерна "наблюдатель".

**Вывод: :** получил навыки работы с файловыми потоками в С#.