Итератор это инумератор (нотация майкросовт)

Коллекция реализовывает в себе (банкомат – хранение денег, кассира (IEnumerator))

using System;

using System.Collections;

namespace InterIEnumerable

{

// Класс UserCollection коллекция (набор) объектов класса Element.

// Для применения foreach, необходимо, чтобы класс реализовывал интерфейс - IEnumerable.

public class UserCollection : IEnumerable, IEnumerator

{

public Element[] elementsArray = null;

public UserCollection()

{

elementsArray = new Element[4];

elementsArray[0] = new Element("A", 1, 10);

elementsArray[1] = new Element("B", 2, 20);

elementsArray[2] = new Element("C", 3, 30);

elementsArray[3] = new Element("D", 4, 40);

}

// Указатель текущей позиции элемента в массиве.

int position = -1;

// ------------------------------------------------------------------------------------------------------------------

// Реализация интерфейса IEnumerator.

// Передвинуть внутренний указатель (position) на одну позицию.

public bool MoveNext()

{

if (position < elementsArray.Length - 1)

{

position++;

return true;

}

else

{ Reset();

return false;

}

}

// Установить указатель (position) перед началом набора.

public void Reset()

{

position = -1;

}

// Получить текущий элемент набора.

public object Current

{

get { return elementsArray[position]; }

}

// -----------------------------------------------------------------------------------------------------------------

// Реализация интерфейса - IEnumerable.

IEnumerator IEnumerable.GetEnumerator()

{

return this as IEnumerator;

}

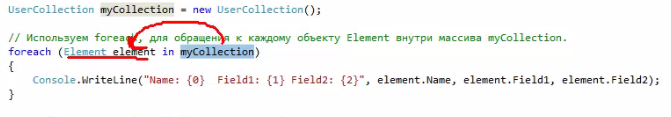
Чтобы данный метод увидеть необходимо сделать приведение к этому типу, т.к. мы указали тип интерфейса IEnumerable.GetEnumerator()



}

}

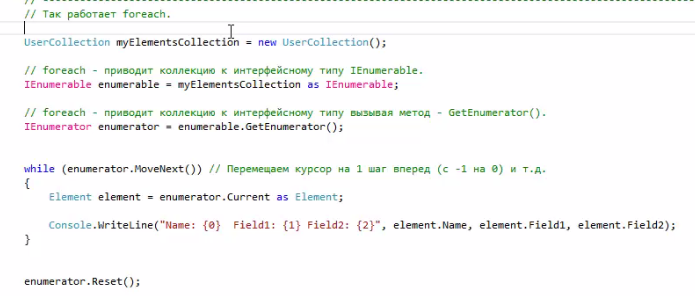
Переменная итерации element, в данном случае мы поочередно извлекаем элементы и помещаем их в переменную итерации



Работа foreach

Foreach Не вызывает самостоятельно метод Reset()

GetEnumerator возвращает значение базового интерфейсного типа IEnumerator

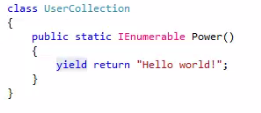


Yield – выход изделий, возвращать, выдавать (в переводе)

Оператор автоматической генерации программного кода итераторов

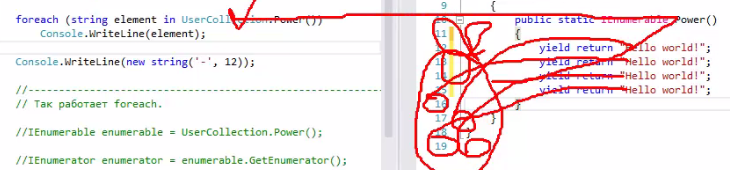
Создает итератор

Без наследования интерфейсов

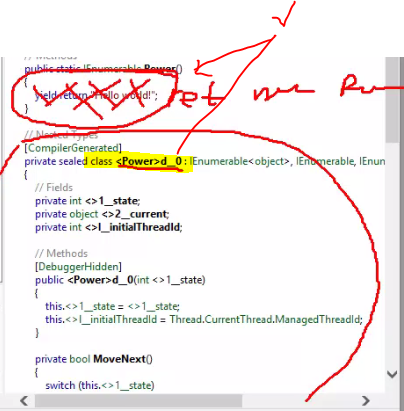




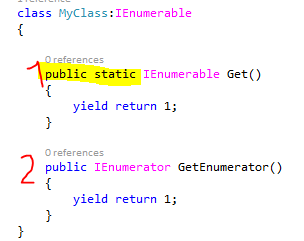
Каждый yield забрасывает себя в мешок, последний связывает его и выкидывает из метода, мешок полный string

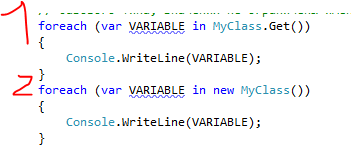


Yield создает вложенный класс и его возвращает, в данном случае (class <Power>d\_0)



Вызывается (Отличия)





Полная работа yield, если заменить данное ключевое слово на класс

using System;

using System.Threading;

using System.Collections;

using System.Collections.Generic;

namespace Yield

{

class UserCollection

{

public static IEnumerable Power()

{

return new ClassPower(-2); этот класс возвращается

}

private sealed class ClassPower : IEnumerable<object>, IEnumerator<object>, IEnumerator, IDisposable

{

// Поля.

private int state;

private object current;

private int initialThreadId;

// Конструктор.

public ClassPower(int state)

{

this.state = state;

this.initialThreadId = Thread.CurrentThread.ManagedThreadId;

}

//private bool IEnumerator.MoveNext() // Так в Рефлекторе

bool IEnumerator.MoveNext()

{

switch (this.state)

{

case 0:

this.state = -1;

this.current = "Hello world!";

this.state = 1;

return true;

case 1:

this.state = -1;

break;

}

return false;

}

IEnumerator<object> IEnumerable<object>.GetEnumerator()

{

if ((Thread.CurrentThread.ManagedThreadId == this.initialThreadId) && (this.state == -2))

{

this.state = 0;

return this;

}

return new UserCollection.ClassPower(0);

}

IEnumerator IEnumerable.GetEnumerator()

{

// Так в Рефлекторе

//return this.System.Collections.Generic.IEnumerable<System.Object>.GetEnumerator();

return (this as IEnumerable<object>).GetEnumerator();

}

void IEnumerator.Reset()

{

throw new NotSupportedException();

}

void IDisposable.Dispose()

{

}

// Свойства.

object IEnumerator<object>.Current

{

get

{

return this.current;

}

}

object IEnumerator.Current

{

get

{

return this.current;

}

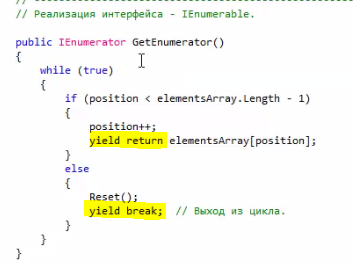
}

}

}

}

Реализация одного интерфейса IEnumerable для коллекции



Если реализовываем обобщенную коллекция то в методе IEnumerator IEnumerable.GetEnumerator() можно вернуть return this as IEnumerator<T>; либо return GetEnumerator();

Но правильней return (this as IEnumerable<T>).GetEnumerator();

class CollectionString<T>:IEnumerable<T>

{

private T str;

public CollectionString(T s)

{

str = s;

}

public IEnumerator<T> GetEnumerator()

{

yield return str;

}

IEnumerator IEnumerable.GetEnumerator()

{

return (this as IEnumerable<T>).GetEnumerator();

}