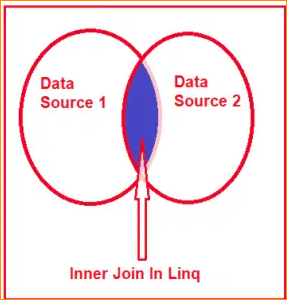
1. What is LINQ Inner Join in C#?

LINQ Inner Join is used to return only the matching elements from both data sources while the non-matching elements are removed from the result set. So, if you have two data sources, and when you perform the Inner Join, then only the matching elements which exist in both data sources are included in the result set. For a better understanding of Inner Join, please have a look at the following image which shows the pictorial representation of Inner Join.

Note: While performing the Inner Join, there should exist a common element or property in both data sources.



1. How to Implement Inner Join in LINQ?

To implement LINQ Inner Join we need to use the Join method in C#. The LINQ Join Method operates on **two data sources or two sequenc**es or you can also **say two collections** such as **inner collection and outer collection.** The Join Method returns a new collection that contains data from both collections and it is the same as the SQL join. There are two overloaded versions of the Join method available in LINQ, their signature is as follows.

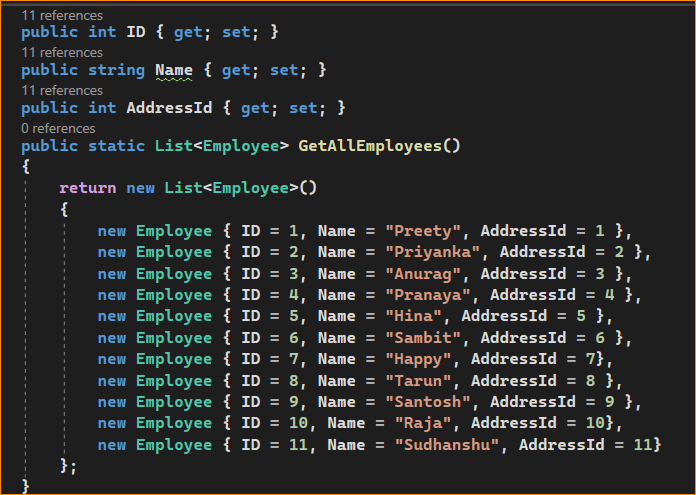


As you can see there are two overloaded versions of the Join Method available in LINQ to perform Inner Join Operations in C#. The one and only difference between the above two overloaded versions is that the second overloaded version takes a comparer as an extra parameter. So, while working with LINQ Join Method (with LINQ Method Syntax) or join operator (with LINQ Query Syntax), we need to understand the following five things.

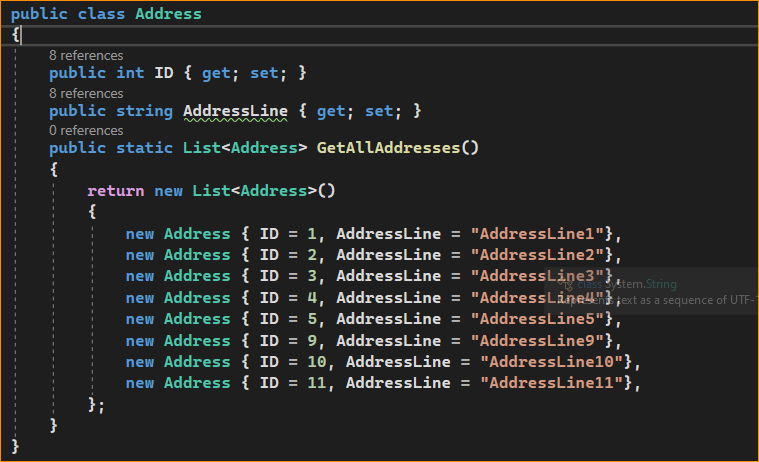
* Outer Data Source: This is the first data source or first collection to be involved in the Join Operation.
* Inner Data Source: This is the second data source or second collection to be involved in the Join Operation.
* Outer Key Selector: This will be the common key in the outer data source.
* Inner Key Selector: This will be the common key in the inner data source.
* Result Selector: Project the data into a result set which will include the properties from both Inner and Outer Data Sources.

Let us understand How to Implement LINQ Inner Join with some examples using C# language. For this, we are going to use the following two model classes

i.e. **Employee and Address**. So, create a class file with the name Employee.cs and then copy and paste the following code into it. This is a very simple class having 3 properties i.e. **Id, Name, and AddressId**. We have also created one method which is going to return a collection of Employees which is going to be our first data source going to be used in the Inner Join.



Now, create another class file with the name Address.cs and then copy and paste the following code into it. This is a very simple class having 2 properties i.e. Id, and AddressLine. We have also created one method which is going to return a collection of addresses which is going to be our second data source going to be used in the Inner Join.



Note: In Real-Time Applications, you need to fetch the data from a database. Here we are not going to focus on how to fetch the data from a database rather we are going to focus on how to perform the inner join. So, here we created the required data sources (i.e. list of employees and addresses) with the hard-coded data.

Here in both data sources, the common property is the Address id i.e. the AddressId property of the Employee data source and the ID property of the Address data source is the common property. As you can see we have 11 records in the employee data source and 8 records in the addresses data source. Further, if you notice some of the data are present in both the data sources.

1. Using LINQ Join Method in C#:

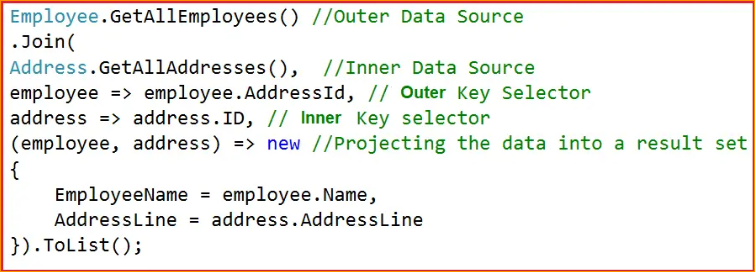
Let us see how to use the LINQ Join Method in C#. Our requirement is to fetch the employee’s name and corresponding address in an anonymous type. But here we need to fetch only the employees which are having the address. If one employee does not have the corresponding address, then we don’t want that employee in our result set. So, basically, we need to perform Inner Join in between the Employee and Address Data Sources. In this case, you can take any data source as the Outer data source as well as any data source as the Inner Data Source.

The following code shows how to perform LINQ Inner Join in C# using the Method Syntax. Here, you can see,

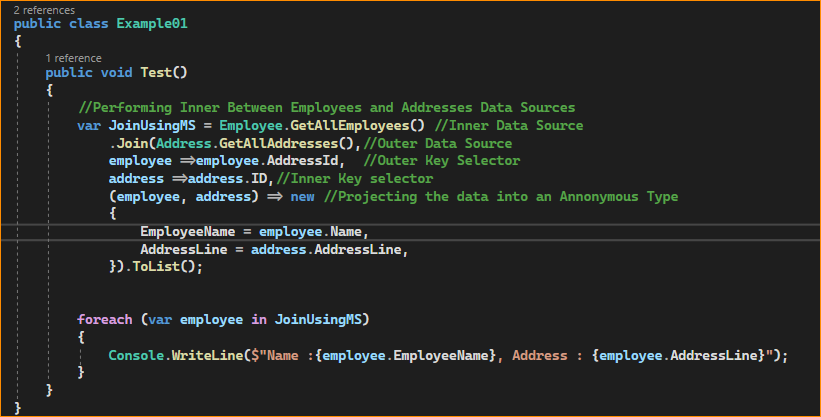
we are using the Join Method**. Employee.GetAllEmployees() is our Outer Data Source** and **Address.GetAllAddresses() is our Inner Data Source.**

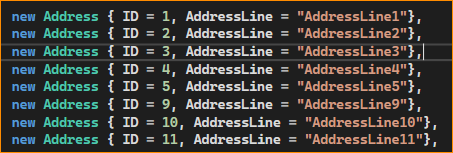
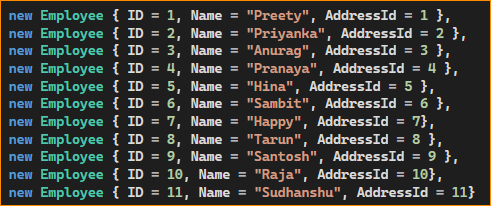
Here, we are **accessing the employee’s information using the employee variable** and **accessing the addresses using the address variable.** Further, you can notice we are specifying the **outer key selector as AddressId using Lambda Expression** and **specifying the Inner Key Selector as ID using the lambda expression**.

These **Outer and Inner Key Selectors are nothing but the common properties in both data sources.** Finally, we are projecting the result to an anonymous type and fetching the employee name to the **EmployeeName property and AddressLine to the AddressLine property**.

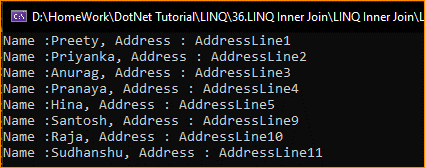


The complete example code is given below. Once the Join Operation is done, then you can access the elements using a for each loop which is shown in the below example. The following code is self-explained, so please go through the comment lines.

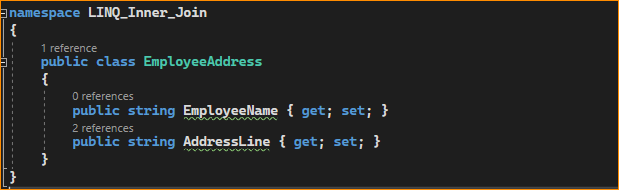




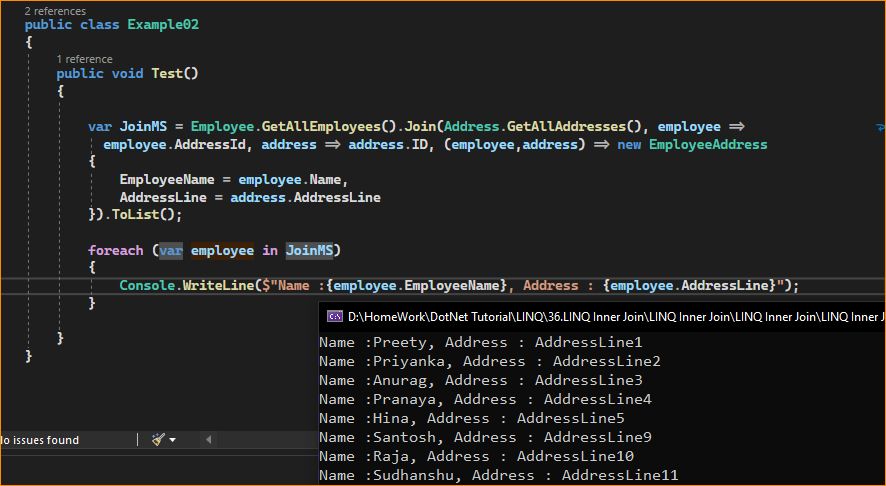
Result-



As you can see in the above output, it only fetches the matching records from both data sources. Instead of projecting the result to an anonymous type, can we project the result to a named type? Yes, it is possible. Let us see how we can do this. First, create a class file with the name EmployeeAddress.cs with the required properties that you want in the result set. As per our requirement, we have created the class with two properties.



Next, modify the Main method of the Program class as follows. Here, you can see, we are projecting the result to the above-created EmployeeAddress type. Further, we have changed the Inner and Outer Data Source order to make sure we are going to get the output as expected when performing the Inner Join Operations in LINQ. Here, we are making the Addresses the Outer Data Source and Employees as the Inner Data Source.



Query Syntax-

LINQ provides the join operator to perform the join using Query syntax in C#. Performing the join operation using query syntax is very much similar to performing the join in SQL. I personally preferred to use Query Syntax instead of Method Syntax as it is syntactically similar to SQL Join.

