LINQ (language-integrated query)

<table>
<tr><td>from clause</td><td><b>from <i>“range var”</i></b> [As <b><i>type</i></b>] <b>in <i>IEnumerable[&lt;T&gt;]</i></b></td></tr>
<tr><td>where clause</td><td><b>where <i>“bool expr”</i></b> | <b><i>Func&lt;TSource,Boolean&gt;</i></b></td></tr>
<tr><td>select clause</td><td><b>select <i>“range var”</i></b> | <b><i>“var member”</i></b> | <b><i>anonymous var</i></b></td></tr>
<tr><td>group clause</td><td><b>group <i>“range var”</i></b> | <b><i>“variable member”</i> by <i>variable</i></b></td></tr>
<tr><td>into</td><td>[<b>group…</b> |<b>join…</b> |<b>select …</b>] <b>into <i>variable</i></b></td></tr>
<tr><td>orderby clause</td><td><b>orderby <i>var</i></b> [<b>ascending</b>|<b>descending</b>] [, <b><i>variable…</i></b>]</td></tr>
<tr><td>join clause</td><td><b>join <i>var</i> in <i>IEnumerable[&lt;T&gt;]</i> on <i>var</i></b>$_i$ <b>equals <i>var</i></b>$_j$</td></tr>
<tr><td>let clause</td><td><b>let <i>“range variable”</i></b> = <b><i>expression</i></b></td></tr>
</table>

examples

```
int[] numbers = { 5, 4, 1, 3, 9, 8, 6, 7, 2, 0 };
var lowNums = from num in numbers
            where num < 5
            select num;
```

```
ContactInfo cInfo = (from ci in contactList
                    where ci.ID == id
                    select ci)
                    .FirstOrDefault();
```

```
IEnumerable<Student> studentQuery1 =
                from student in app.students
                where student.ID > 111
                select student;
```

```
IEnumerable<String> studentQuery2 =
    from student in app.students
    where student.ID > 111
    select student.Last;
```

```
IEnumerable<IGrouping<char,Student>> studentQuery1 =
    from student in students
    group student by student.Last[0];
```

```
var studentQuery2 =
    from student in students
    group student by student.Last[0] into g
    orderby g.Key
    select g;
```

```
var booleanGroupQuery =
    from student in students
    group student by student.Scores.Average() >= 80;
```

```
var studentQuery =
   from student in students
   let avg = (int)student.Scores.Average()
   group student by (avg == 0 ? 0 : avg / 10) into g
   orderby g.Key
   select g;
```

```
IEnumerable<Student> sortedStudents =
   from student in students
   orderby student.Last, student.First
   select student;
```

```
var innerJoinQuery =
    from category in categories
    join prod in products on category.ID equals
prod.CategoryID
    select new { ProductName = prod.Name, Category =
category.Name }; //produces flat sequence
```

```
var innerGroupJoinQuery =
    from category in categories
    join prod in products on category.ID equals
        prod.CategoryID into prodGroup
    select new { CategoryName = category.Name,
        Products = prodGroup };
```