بِسْمِ اللهِ الرَحْمنَ الرَحیمِ

آموزش سی‌شارپ – بخش ۴۳ ← استفاده از List <>

- **مقدمه**

آرایه‌ها، یک مفهوم بسیار مهم در برنامه‌نویسی می‌باشند؛ بدین صورت که همانند یک مجموعه، مجموعه‌ای از اطلاعات را در درون خود نگه‌داری می‌کنند. اما آیا به یکی از مشکلات آن دقت کرده‌اید؟ اگر هیچ وقت ندانیم که سایز آرایه‌ی ما چقدر باید باشد، چه کار باید انجام دهیم؟ شاید بگویید که در اول اجرای برنامه از کاربر سوال کنیم، اما اگر خود کاربر حتی نداند چه تعدادی آیتم دارد چه؟ مثلاً فرض کنید که یک برنامه‌ی دفترچه تلفن نوشته‌اید؛ نه کاربر و نه شما می‌دانید که چه تعداد مخاطبین می‌خواهید اضافه کنید و قطعاً در آینده و روز به روز افزایش پیدا خواهد کرد.

پس باید به مفهومی به نام List<> مراجعه کنیم. یکی از مشکلاتی که توسط List<> حل می‌شود، همین مشکل است، پس با ما باشید.

- **استفاده از List<>**

همانطور که از نامش پیداست، مفهومی که با آن سر و کار داریم یک «لیست» است؛ یعنی زمانی از آن استفاده می‌کنیم که بخواهیم آیتم جدیدی به آن اضافه، آیتمی را از آن حذف و یا در آن جستجو کنیم؛ یا مثلاً اطلاعات وارد شده را Sort (مرتب) کنیم و ...

برای مثال، شما معلمی هستید که در کلاس خود ۱۰ دانش‌آموز حضور دارند. شما می‌توانید لیستی از دانش‌آموزان خود درست کنید که اسامی آن‌ها را ذخیره کرده، اگر آن‌ها را بر اساس حروف الفبا مرتب کنید، دانش‌آموز جدیدی را اضافه کنید و یا دانش‌آموزشی را از آن حذف نمایید. اگر از یک آرایه معمولی استفاده کنید، می‌دانید که انجام کارهای فوق بسیار سخت است.

اکنون Visual Studio را اجرا نموده و یک پروژه جدید ایجاد نمایید. یک Button و Listbox اضافه کنید. بر روی Button خود دابل-کلیک کنید تا وارد بخش کدنویسی مربوطه شوید. به کدهای آماده (خصوصاً کدهای بالا) دقت کنید. الان شما کدهایی را می‌بینید که با Using شروع می‌شوند.

Using مربوط به List<>، همانی هست که کد آن بدین شکل است: System.Collections.Generic

اگر نمی‌توانید آن را ببینید، خودتان آن را بصورت دستی اضافه کنید.

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Text;
using System.Windows.Forms;

namespace collections
{
    public partial class Form1 : Form
    {
        public Form1()
        {
```

نحوه نعریف یک لیست در سی‌شارپ به‌شکل زیر می‌باشد:

```
List<string> students = new List<string>();
```

ابتدا با کلمه کلیدی List شروع کرده، در داخل علامت‌های مربوطه نوع لیست خودتان را انتخاب می‌کنید و پس از ایجاد یک فاصله (Space)، نام متغیر را تعیین می‌کنید و در نهایت، آن را new می‌کنید. کدهای شما می‌بایست که به شکل زیر شده باشند:

```
namespace lists
{
    public partial class Form1 : Form
    {
        public Form1()...

        private void button1_Click(object sender, EventArgs e)
        {

            List<string> students = new List<string>();

        }
    }
}
```

اکنون زمان اضافه کردن اطلاعات به لیست خود می‌باشید. برای اضافه کردن اطلاعات به لیست خود، می‌توانید پس از ذکر نام لیست خود که انتخاب کرده‌اید، از متد Add استفاده نمایید. به کدهای زیر دقت کنید:

```
students.Add("Jenny");

students.Add("Peter");

students.Add("Mary Jane");
```

کدهای شما اکنون بدین شکل خواهند بود:

```
private void button1_Click(object sender, EventArgs e)
{

    List<string> students = new List<string>();

    students.Add("Jenny");
    students.Add("Peter");
    students.Add("Mary Jane");

}
```

برای دسترسی به لیست خود نیز می‌توانید از حلقه foreach استفاده کنید. کد زیر را اضافه کنید:

```
foreach (string child in students)

{

        listBox1.Items.Add(child);

}
```

بدین ترتیب که ما با استفاده از این حلقه در این لیست در حال گردش هستیم. اگر هم می‌خواهید از لحقه معمولی For استفاده کنید، می‌توانید از کد زیر استفاده نمایید:

```
for (int = 0; i <students.Count; i++)

{

        listBox1.Items.Add(students[i]);

}
```

دقت کنید که Count در student.Count تعداد آیتم‌های موجود در لیست را برمی‌گرداند.

اما پیشنهاد می‌شود که در لیست‌ها از حلقه‌های Foreach استفاده کنید.

کدهای خودتان را اجرا کنید؛ نتیجه را خواهید دید!

مرتب‌سازی لیست

یکی از مهمترین کارهایی که می‌توان در List<> انجام داد، مرتب‌سازی آن می‌باشد. شما می‌توانید با استفاده از متد Sort()، اقدام به مرتب‌سازی مقادیر موجود در داخل لیست اقدام نمایید. به کد زیر دقت کنید:

```
students.Sort();
```

اگر کد زیر را پیش از نمایش آیتم‌ها در Listbox اضافه کنید، خواهید دید که نتیجه‌ی حاصل، یک لیست مرتب‌شده بر اساس حروف الفبا می‌باشد. اگر نوع لیست را از نوع عدد می‌گرفتید، نتیجه‌ی حاصل، بر اساس ترتیب اعداد می‌بود.

برای اینکه لیست خود برعکس کنید، می‌توانید از متد Reverse() استفاده کنید. به کد زیر دقت کنید:

```
stundents.Reverse();
```

حذف مقدار از List<>

شما می‌توانید از دو روش مقادیر موجود در لیست را حذف کنید؛ یا اینکه مقدار مورد نظر را می‌دانید و آن را به تابع می‌گویید تا آن را حذف کند، و یا یک رنجی از می‌خواهید حذف کنید که باید شماره محل قرار گرفتن آن را به متد پاس دهید. این کارها توسط دو متد Remove و RemoveRange انجام می‌شود. به کدهای زیر دقت کنید:

```
student.Remove(“Peter”);
```

این کد، آیتم مربوط به Peter را حذف می‌کند.

```
student.RemoveRange(0, 2);
```

این کد، اولین مقدار (که در اینجا صفر است) محل شروع حذف کردن را به شما می‌گوید و مقدار دوم (که ۲ می‌باشد) تعداد خانه‌های بعد از اولین مقدار (که اینجا صفر است) را که می‌خواهید حذف کنید را از شما می‌گیرد.

لینک ثابت مقاله (دسترسی آنلاین): http://www.gooyait.com/1392/05/10/c-sharp-tutorial-part-43.html