لیست پیوندی

لیست‌های پیوندی (**Linked Lists**) در زبان برنامه‌نویسی سی‌شارپ یکی از ساختارهای داده‌ای مهم هستند که برای ذخیره‌سازی مجموعه‌ای از عناصر به صورت داینامیک و غیرمستقیم استفاده می‌شوند. بر خلاف آرایه‌ها که اندازه ثابت دارند، لیست‌های پیوندی به ما این امکان را می‌دهند که به راحتی عناصر جدیدی را اضافه یا حذف کنیم. هر عنصر در لیست پیوندی، به یک عنصر دیگر اشاره می‌کند که به آن " گره" (**Node**) گفته می‌شود. گره‌ها شامل دو قسمت هستند: داده و اشاره‌گر به گره بعدی. این ساختار به ما اجازه می‌دهد تا به راحتی عملیات‌هایی مانند اضافه کردن، حذف کردن و جستجو کردن عناصر را انجام دهیم

پیاده سازی لیست پیوندی

برای پیاده سازی یک لیست پیوندی از کلاس **LinkedList** استفاده خواهیم کرد:

خصوصیات کلاس

این کلاس دارای چندین **property** است که به مدیریت و دسترسی به عناصر لیست پیوندی کمک می‌کند. یکی از این خصوصیات **First** است که به اولین گره در لیست اشاره می‌کند و می‌تواند برای دسترسی سریع به اولین عنصر استفاده شود. **Last** نیز به آخرین گره در لیست اشاره دارد و به ما این امکان را می‌دهد که به راحتی به آخرین عنصر دسترسی پیدا کنیم. همچنین، **Count** تعداد عناصر موجود در لیست را باز می‌گرداند.

متدهای کلاس

متد **AddFirst** برای اضافه کردن یک عنصر جدید به ابتدای لیست استفاده می‌شود، در حالی که **AddLast** برای افزودن یک عنصر به انتهای لیست به کار می‌رود. متد **AddBefore** و **AddAfter** به ما این امکان را می‌دهند که یک عنصر را قبل یا بعد از یک گره مشخص اضافه کنیم. برای حذف عناصر، می‌توان از متد **Remove** برای حذف یک گره خاص و **RemoveFirst** و **RemoveLast** برای حذف اولین یا آخرین گره استفاده کرد. متد **Clear** نیز برای پاک‌سازی تمام عناصر لیست کاربرد دارد. علاوه بر این، متد **Contains** برای بررسی وجود یک عنصر در لیست و متد **Find** برای جستجوی یک گره خاص به کار می‌روند.

مثال : برنامه ای بنویسید که نام دانشجویی را از کاربر گرفته و در لیستی پیوندی بنام **students** قرار دهد.

Program.cs

```
LinkedList<string> students=
new LinkedList<string>();
string name=Console.ReadLine();
students.AddFirst(name);
```

مثال : با فرض موجود بودن لیستی پیوندی بنام **students** برنامه ای بنویسید که داده های موجود درلیست را در خروجی نمایش دهد.

Program.cs

```
LinkedList<string> students=
new LinkedList<string>();
// .....
foreach(string name in students){
    Console.WriteLine(name);
}
```

مثال : برنامه ای بنویسید که ده عدد تصادفی تولید و به لیستی پیوندی بنام **nums** اضافه نماید.

```
Program.cs
LinkedList<int> nums=
new LinkedList<int>();
LinkedListNode<int> nd;
Random r=new Random();
nums.AddFirst(r.Next(10,99));
nd=nums.First;
for (int i=0;i<9;i++){
    nums.AddAfter(nd,r.Next(10,99));
}
```

مثال : کلاسی بنام **Test** بنویسید که شامل متدی بنام **generateNums** باشد ، این متد ده عدد تصادفی دورقمی تولید کرده و در لیستی پیوندی بنام **nums** که در این کلاس بصورت **private** تعریف شده است قرار دهد.

```
Test.cs > ...
class Test{
private LinkedList<int> nums=
new LinkedList<int>();
private LinkedListNode<int> nd;
public void generateNums(){
Random r=new Random();
nums.AddFirst(r.Next(10,99));
nd=nums.First;
for (int i=0;i<9;i++){
    nums.AddAfter(nd,r.Next(10,99));
}}}
```

مثال : به کلاس **Test** متدی بنام **showCount** اضافه نمایید که عددی را بعنوان پارامتر گرفته و تعداد آن عدد در **nums** را بازگرداند.

```
class Test
    public int showCount(int n){
      int count=0;
      foreach(int i in nums){
        if (i==n){count++;}
      }
      return count;
    }
```

مثال : به کلاس **Test** متدی بنام **addToLIst** اضافه نمایید که عددی را بعنوان پارامتر گرفته و در صورت عدم وجود عدد آن را به لیست اضافه نماید.

```
class Test
    public void AddToList(int n){
      if(showCount(n)==0){
       nums.AddFirst(n);
      }
    }
```

نکته : ما در این مثال از متد نوشته شده در مثال قبلی استفاده کردیم.

مثال : به کلاس **Test** متدی بنام **showMax** اضافه نمایید که بزرگترین عدد موجود در **nums** را بازگرداند.

```
class Test
      public int showmax(){
       return nums.Max();
     }
```

تمرین ۱ : به کلاس **Test** متدی بنام **removeRepeat** اضافه نمایید که تمامی تکرار های لیست **nums** را حذف و لیست را بازگرداند.

تمرین ۲ : به کلاس **Test** متدی بنام **maxCount** اضافه نمایید که عددی را بعنوان پارامتر گرفته و تعداد عدد های بزرگتر از آن موجود در لیست را بازگرداند.

تمرین ۳ : به کلاس **Test** متدی بنام **findNum** اضافه نمایید که عددی را بعنوان پارامتر گرفته و در صورت وجود عدد **True** و در غیر اینصورت **False** را بازگرداند.

تمرین ۴ : به کلاس **Test** متدی بنام **rNums** اضافه نمایید تا لیست موجود در کلاس بنام **nums** را بصورت معکوس شده (از آخر به اول) بازگرداند.