دانشکده‌ی مهندسی کامپیوتر
تمرین‌های سری سوم برنامه‌سازی پیشرفته

مدرس: سید صالح اعتمادی
طراحی و تدوین: امیرمحمد احساندار، محمدمهدی عبدالله‌پور، مبینا کاشانیان، یگانه مرشدزاده

مهلت ارسال:
شنبه ۱۸ اسفند ۹۷

فهرست مطالب

۱ مقدمه

این تمرین شما شامل سه مرحله است که برای انجام آن لازم است به اندازه کافی وقت بگذارید تا به نتیجه مطلوب برسید، فراموش نکنید که هر چه در این درس سرمایه‌گذاری کنید، در مراحل بالاتر قوی‌تر عمل خواهید کرد.

۱.۱ موارد مورد توجه

- توجه داشته باشید که برای کسب نمره‌ی قبولی درس کسب حداقل نصف نمره‌ی هر سری تمرین الزامی می‌باشد.
- مهلت ارسال پاسخ تمرین تا ساعت ۲۳:۵۹ روز اعلام‌شده است. توصیه می‌شود نوشتن تمرین را به روزهای نهایی موکول نکنید. استثنائا به دلیل انتشار دیر هنگام این تمرین دو روز به عنوان مهلت اضافی بدون کسر نمره در نظر گرفته شده است که شما می‌توانید از این دو روز تاخیر برای این تمرین یا تمرین دیگری استفاده کنید.
- هم‌کاری و هم‌فکری شما در حل تمرین مانعی ندارد، اما پاسخ ارسالی هر کس حتما باید توسط خود او نوشته شده باشد.
- مبنای درس، اعتماد بر پاسخ ارسالی از سوی شماست؛ بنابراین ارسال پاسخ در ریپازیتوری گیت شما به این معناست که پاسخ آن تمرین، توسط شما نوشته شده است. در صورت تقلب یا اثبات عدم نوشتار پاسخ حتی یک سوال از تمرین، برای هر دو طرف تقلب‌گیرنده و تقلب‌دهنده نمره‌ی مردود برای درس در نظر گرفته خواهد شد.
- توجه داشته باشید که پاسخ‌ها و کدهای مربوط به هر مرحله را بایستی تا قبل از پایان زمان مربوط به آن مرحله، در سایت Azure (طبق توضیحات کارگاه‌ها و کلاس‌ها) بفرستید. درست کردن `Pull request` و `Complete` کردن `Pull request` و انتقال به شاخه‌ی `master` پس از تکمیل تمرین فراموش نشود!
- پس ازپایان مهلت ارسال تا ۲ روز به ازای هر روز تاخیر ۱۰ درصد از نمره مربوط به تمرین کسر خواهد شد و پس از ۲ روز نمره‌ای به تمرین تعلق نخواهد گرفت.
- برای طرح سوال و پرسش و پاسخ از صفحه درس در Quera استفاده کنید.

۲ آشنایی با رشته و فایل

۱.۲ تنظیمات اولیه و آماده‌سازی

در این بخش شما قرار است ویژگی‌ها و متدهایی که از رشته `string` و فایل `File` که به تازگی یاد گرفته‌اید را تمرین کنید. یک `Solution` جدید به نام `A1S1` در ریشه `Git` خود بسازید. در پروژه `A1S1` فایل ضمیمه شده `Program.cs` را پس از حذف `Program.cs` قبلی به پروژه اضافه (`Add`) کنید. سپس یک پروژه تست بسازید. توجه داشته باشید که نام پیشفرض پروژه و متد تست `TestMethod` نوشته شود تغییر داده نشود! نکته بسیار مهم درباره این تمرین این است که برای تمام متدهایی که طراحی می‌کنید، یک پیاده‌سازی تست‌ها از اهمیت بسیار بالایی برخوردار خواهد بود، پس برای این قسمت به طور ویژه وقت در نظر بگیرید)

۱.۱.۲ `CaculateLength`

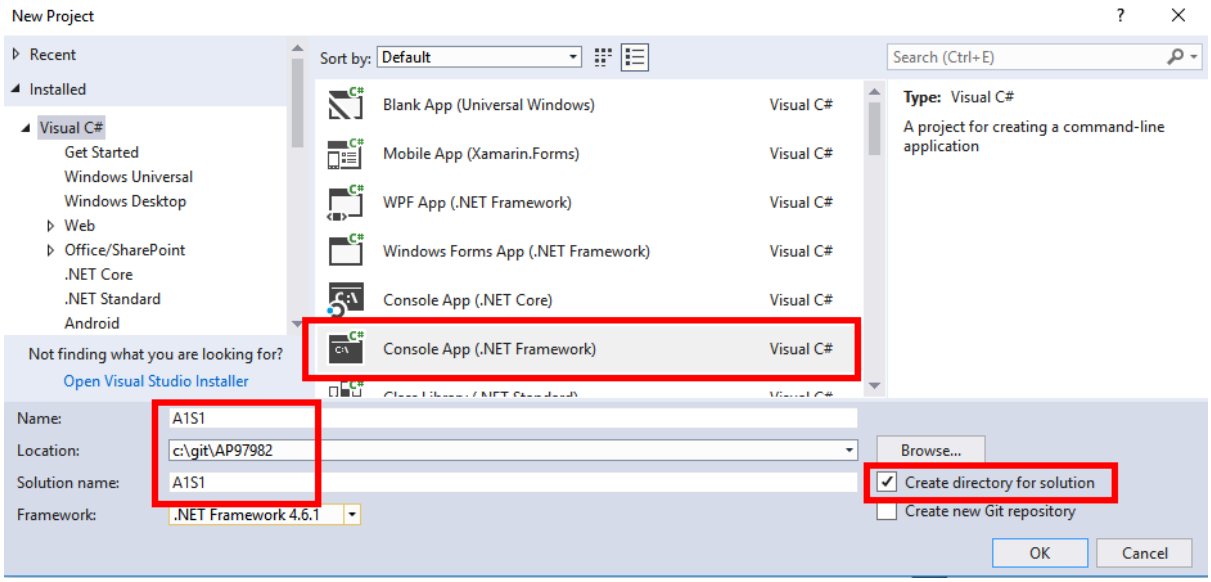
متد `CaculateLength` را پیاده‌سازی کنید طوری که یک رشته به عنوان پارامتر ورودی دریافت کند و طول آن رشته را برگرداند.

خروجی نمونه	ورودی نمونه
21	`this is a test string`

۲.۱.۲ `LetterCount`

متد `LetterCount` را پیاده‌سازی کنید طوری که یک رشته ورودی گرفته و تعداد حروف آن را محاسبه کرده و برگرداند. برای پیاده‌سازی میتوانید از متد `char.IsLetter()` استفاده کنید. اطلاعات بیش‌تر در این لینک

خروجی نمونه	ورودی نمونه
16	من همیشه سر وقت هستم



شکل ۱: ایجاد پروژه جدید

۳.۱.۲ LineCount

متد `LineCount` را پیاده‌سازی کنید طوری که یک رشته ورودی دریافت کند و تعداد خطوط (کاراکترهای `'\n'`) را برگرداند.
اطلاعات بیش‌تر در این لینک

۴.۱.۲ FileLineCount

متد `FileLineCount` را پیاده‌سازی کنید طوری که آدرس کامل یک فایل را به عنوان پارامتر ورودی دریافت کند و تعداد خطوط آن فایل را برگرداند. برای تست کردن متد `FileLineCount` از قطعه کد زیر استفاده کنید. این کد مسیر کامل یک فایل به همراه تعداد خطوط آن را برمی‌گرداند.

```
۱  private static string GetTestFile(out int lineCount, out int charCount)
۲  {
۳      charCount = 0;
۴      string tmpFile = Path.GetTempFileName();
۵      lineCount = new Random(0).Next(10, 100);
۶      List<string> lines = new List<string>();
۷      for(int i=0; i< lineCount; i++)
۸      {
۹          string line = $"Line number {i}";
۱۰         charCount += line.Length;
۱۱         lines.Add(line);
۱۲     }
۱۳     File.WriteAllLines(tmpFile, lines);
۱۴     return tmpFile;
۱۵ }
```

اطلاعات بیشتر در این لینک
اگر در مورد آدرس نسبی و کامل ابهامی دارید به پیوست ۱.۵ در همین سند مراجعه نمایید.

۵.۱.۲ ListFiles

متد `ListFiles` را پیاده‌سازی کنید طوری که آدرس کامل یک پوشه را به عنوان پارامتر ورودی دریافت کند و لیست فایل‌های درون پوشه را به‌صورت `string[]` برگرداند.

```
۱  private static string[] GetTestDir(out string tmpDir)
```

```
{
   tmpDir = Path.GetTempFileName();
   if (File.Exists(tmpDir))
      File.Delete(tmpDir);

   if (!Directory.Exists(tmpDir))
      Directory.CreateDirectory(tmpDir);
   else
      foreach (string file in Directory.GetFiles(tmpDir))
         File.Delete(file);

   int rndNum = new Random(0).Next(10, 20);
   List<string> files = new List<string>();
   for (int i=0; i<rndNum; i++)
   {
      string fileName = Path.Combine(tmpDir, $"file{i}.txt");
      File.WriteAllText(fileName, $"file{i}.txt content");
      files.Add(fileName);
   }
   return files.ToArray();
}
```

۶.۱.۲ FileSize

متد `FileSize` را پیاده‌سازی کنید طوری که آدرس کامل یک فایل را از ورودی گرفته و حجم فایل را بر حسب تعداد کاراکترهای موجود در فایل برگرداند.

به این صورت که شما در ابتدا تعداد کاراکترهای موجود در یک فایل را به دست می‌آورید و خروجی را بر حسب تعداد کاراکتر برمی‌گردانید. برای تست این متد از متد `GetTestFile` استفاده کنید.

۳ پیاده‌سازی و آدرس‌دهی

یک برنامه اجرایی پیاده‌سازی کنید که که آدرس کامل یک پوشه را به عنوان پارامتر (`command line argument`) دریافت کند و به صورت بازگشتی جمع حجم فایل‌های موجود در آن پوشه و تمام زیرپوشه‌ها را (به صورت بازگشتی) محاسبه کرده و برگرداند. برای گرفتن حجم فایل می‌توانید از `FileInfo` استفاده کنید. برای این پروژه مشابه آنچه در شکل ۱.۲ نشان داده شده، پروژه ای به نام `A1S2` درست کنید و تمام برنامه و متدهای لازم را در همان `Program.cs` که به طور پیش فرض همراه با ساخت پروژه ایجاد می‌شوند پیاده‌سازی کنید. برای این قسمت نیاز به اضافه کردن پروژه و متدهای تست نیست.

اطلاعات بیش‌تر در این لینک

۴ تحلیل احساسی واکنش کاربران به توییت اشخاص

برای این قسمت از بین شخصیت‌های پرمخاطب فضای مجازی n نفر را انتخاب کردیم و از بین ۲۰ توییت اخیر آنها یک توییت با بیشترین لایک را انتخاب کردیم. ما متن توییت و متن حداکثر ۱۰۰ منشن (پاسخ دیگر کاربران) آخر این توییت را در ساختار خاصی در اختیار شما قرار داده ایم. این توییت‌ها با ساختار زیر در n عدد فایل در دایرکتوری `"TwitterData"` ذخیره شده‌اند و در اختیار شما قرار داده شده است. هدف نهایی جمع‌بندی تحلیل احساسی کاربران به یک توییت از هر کدام از این شخصیت‌ها است.

ابتدا برای این پروژه مشابه آنچه در شکل ۱.۲ نشان داده شده، پروژه ای به نام `A1S3` درست کنید.

```
TwitterData\Tweets\<screenName>.txt
<tweet text without media>
<1st reply without media>
<2nd reply without media>
<3rd reply without media>
.
.
.
<100th reply without media>
```

۱.۴ بررسی دستی داده‌ها

- فایل‌ها را بررسی کنید. برای بررسی فایل‌ها از `Notepad++` استفاده کنید. آیا شخصیت صاحب اکانت توییتر را می‌شناسید؟ اگر نه با استفاده از فیلترشکن مناسب (!) اکانت توییت مورد نظر را بررسی کنید و حدس بزنید توییت کیست. حدس خود را در فایلی به نام `MyGuess.txt` یادداشت کرده و در پوشه `c:\git\AD97982\A1S3` ذخیره کنید. در هر خط ابتدا نام فایل را گذاشته و در مقابل آن شخصیت مورد نظر را بعد از `:` یادداشت کنید.
- متن توییت این فرد را مطالعه کنید. محتوای متن توییت این فرد در کدام دسته‌بندی زیر جای می‌گیرد: شخصی، سیاسی، اجتماعی و.... تقسیم‌بندی خود را در همان فایل بالا در انتهای هر خط بعد از کاراکتر `:` ذخیره کنید.
- واکنش کاربران را نسبت به این توییت‌ها مطالعه کنید. بعد از مطالعه تمام واکنش‌ها به یک توییت جمع‌بندی خود از واکنش کاربران به این توییت را بصورت عددی بین `-۲` تا `+۲` بنویسید. که `-۲` یعنی خیلی منفی و `+۲` یعنی خیلی مثبت. این نتیجه را در همان فایل بالا در انتهای هر خط بعد از کاراکتر `:` ذخیره کنید.

۲.۴ بررسی خودکار

هدف این قسمت این است که نتیجه‌گیری که از واکنش کاربران به توییت‌های افراد گرفتین را به صورت خودکار محاسبه کنیم و در نهایت ارزیابی از این روش محاسبه داشته باشیم. برای این کار نیاز به یک دادگان تحلیل احساسی داریم. برای این منظور از دادگان LexiPers (که این مقاله‌ی منبع، برای کلمات مثبت و منفی در زبان فارسی است) استفاده کرده و فهرستی از کلماتی که بار مثبت و منفی دارند را در فایل‌هایی به نام `positive.txt` و `negative.txt` استخراج کردیم به این صورت که هر یک از لغات در یک خط هستند.

نکته بسیار مهم درباره این تمرین این است که برای تمام متدهایی که طراحی می‌کنید، یک `TestMethod` نوشته شود.

۱. متدی پیاده‌سازی کنید که آدرس کامل یک فایل لغات را از ورودی دریافت کند و آن را به صورت آرایه‌ای از لغات (`string[]`) برگرداند.

```
public static string[] Q1_GetWords(string path)
{
    // Write your code here and remove next line
    return new string[]{};
}
```

۲. متدی پیاده‌سازی کنید که یک آرایه (از لغت) و یک لغت به عنوان پارامتر بگیرد و اگر لغت در لیست لغات بود `true` برگرداند.

```
public static bool Q2_IsInWords(string[] words, string word)
{
    // Write your code here and remove next line
    return false;
}
```

۳. متدی پیاده‌سازی کنید که یک جمله/توییت به عنوان پارامتر ورودی بگیرد و لیست لغات آن را بصورت آرایه از `string` برگرداند. جدا کننده کلمات این کاراکترها هستند: (`"! ? #@',` ...)

```
public static string[] Q3_GetWordsOfTweet(string tweet)
{
    // Write your code here and remove next line
    return new string[]{};
}
```

۴. متدی پیاده‌سازی کنید که یک توییت و دو آرایه لغات مثبت و منفی را به عنوان پارامتر ورودی دریافت کند و بار مثبت/منفی توییت را محاسبه کند. به ازای هر کلمه مثبت $+1$ و به ازای هر کلمه منفی -1 اضافه می‌شود. در نهایت مجموع این اعداد می‌شود بار مثبت/منفی که به صورت یک عدد `Integer` برگردانده می‌شود.

```
public static int Q4_GetPopChargeOfTweet(string tweetstring[] posWords, string[]
    negWords)
{
   // Write your code here and remove next line
   return 0;
}
```

۵. متدی پیاده‌سازی کنید که لیست توییت‌ها را به عنوان آرایه‌ای از `string` دریافت کند و میانگین بار مثبت/منفی آن را برگرداند.

```
public static double Q5_GetAvgPopChargeOfTweets(string[] tweets, string[] negWords,
    string[] posWords)
{
  // Write your code here and remove next line
  return 0;
}
```

۶. بعد از پیاده‌سازی متدهای بالا نوبت به پیاده‌سازی برنامه نهایی است. ابتدا لازم است که پوشه `TwitterData` که همراه پروژه به شما داده شده در پوشه `c:\git\AP97982\A1S3\A1S3` کپی کنید. برای تست اینکه داده ها را در پوشه درست کپی کرده اید لازم است که قطعه کد زیر بدون خطا اجرا شود و لیست فایل ها را در خروجی چاپ کند. حالا با استفاده از متدهایی که در بالا پیاده سازی کرده اید متد `Main` را بگونه ای پیاده سازی کنید که فایلی با نام `result.txt` در پوشه `c:\git\AP97982\A1S3\A1S3` ذخیره کند. لازم است محتوای این فایل شامل یک خط برای هر فایل توییت که در ابتدای هر خط نام فایل و سپس کاراکتر `:` و در ادامه عدد محاسبه شده در متد شماره ۵ در بالا را قرار گیرد.

```
static void Main(string[] args)
{
  string[] files = Directory.GetFiles(@"..\..\TwitterData\Tweets");
  foreach (var file in files)
     Console.WriteLine(file);

  files = Directory.GetFiles(@"..\..\TwitterData\Words");
  foreach (var file in files)
     Console.WriteLine(file);
}
```

برای ارسال این تمرین لازم است علاوه بر فایل‌های `.csproj` ، `.cs` ، `.sln` فایل‌های `.txt` مربوط به محاسبه دستی، خودکار و داده‌های ورودی به گیت اضافه شوند.

۵ پیوست‌ها

۱.۵ آدرس کامل و نسبی

روش استاندارد آدرس‌دهی دارای سه ویژگی اصلی می‌باشد:

۱. مشخص کردن آدرس درایوی که فایل در آن وجود دارد که با علامت `:` (به این علامت‌ها به اصطلاح Directory Separator می‌گویند) آن درایو را مشخص می‌کنیم مانند:

`D:\, C:\, E:\`

۲. آدرس محلی که فایل در آن قرار دارد.

۳. نام فایل

در حالت کلی برای آدرس‌دهی هر فایل می‌توان از دو راه استفاده کرد:

- آدرس مطلق (Absolute path): اگر آدرس‌دهی ما همه‌ی سه ویژگی مذکور را داشته باشد به آن آدرس‌دهی مطلق می‌گوییم در واقع با استفاده از آدرس‌دهی مطلق به صورت کامل و با جزئیات می‌توانیم آدرس آن فایل، نام فایل و در نهایت درایوی که آن فایل در آن موجود است را مشاهده کنیم و آدرس‌دهی از ریشه (Root Directory) ارائه می‌شود (تمام دایرکتوری‌های دیگر در سیستم فایل از دایرکتوری ریشه منشعب می‌شود) و به همین دلیل به صورت منحصر به فرد شناخته می‌شود به طور مثال اگر آدرس‌دهی به صورت زیر باشد به آن مطلق می‌گوییم.

`C:\Documents\HomeWorks\AP_Assignment.pdf`

- آدرس نسبی (Relative path): اگر سه ویژگی استاندارد را ذکر نکنیم برای مثال اگر درایو را مشخص نکنیم و آدرس دهی را بااگر سه ویژگی استاندارد را ذکر نکنیم برای مثال اگر درایو را مشخص نکنیم و آدرس‌دهی را با استفاده از علامت‌های `\` یا `..` یا همان علامت‌های (Directory Separator) انجام دهیم به این نوع آدرس، آدرس نسبی می‌گویند. به طور مثال:

`..\Publications\RelativePath.pdf`

۲.۵ Program.cs

```cs
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace A3
{
   public class Program
   {
      public static void Main(string[] args)
      {
      }

      public int CaculateLength(string str)
      {
         // Write your code here and remove next line
         return 0;
      }

      public int LetterCount(string str)
      {
         // Write your code here and remove next line
         return 0;
      }

      public int LineCount(string str)
      {
         // Write your code here and remove next line
         return 0;
      }

      public int FileLineCount(string filePath)
      {
         // Write your code here and remove next line
         return 0;
      }

      public string[] ListFiles(string dirPath)
      {
         // Write your code here and remove next line
         return new string[] { };
      }

```

```
45      public double FileSize(string filePath)
46      {
47         // Write your code here and remove next line
48         return 0;
49      }
50   }
51 }
```