2014“亚马逊•种子杯”

初赛试题

2014/10/18

1. 题目概述

许多程序设计语言都支持利用正则表达式进行字符串操作。正则表达式的概念最开始是由 Unix 中的工具软件(如 sed 和 grep)普及开的。发展到今天，正则表达式已经用在了很多的高级程序设计语言中，是程序员们喜爱的工具。

本题中，我们要求选手实现一个简单的正则表达式引擎。

2. 基本要求

1) 编程语言：C/C++ (附加库只允许标准库)
   具体允许的库请参见：http://en.cppreference.com/w/，但是带有 (C++11)标志是禁止的。
   编译器：VC/gcc
2) 所有的文本采用 UTF-8 编码。 我们会在 windows vista/7/8 32bit/64bit 几种环境中测试你的程序，在任意一种环境中运行通过即可。如果对运行环境有特殊要求的，请在提交上来的文档中说明。
3) 所完成的功能一律按照详细需求中的标准实现。仅仅要求匹配 ASCII 中的可显示字符。
4) 你的正则表达引擎不需要处理表意冲突的情况。例如 . 和 \w 不会同时用来匹配某个字符。
5) 约定每个测试用例存在多个输出结果时，仅仅匹配第一个输出结果。
6) 约定所有的换行符都使用\n 表示。所有的测试文本中不会出现回车符\r。
7) 此次初赛将会根据测试用例给分，通过一个测试用例即可得到相应的分值。
8) 不要求能识别错误的正则表达式， 但是如果你做了一部分错误处理的工作，请在文档中说明，我们将会酌情加分。
9) 对引擎性能不作特殊要求。如果你在性能优化上做了处理 ,同样请在文档中说明 ,我们将会酌情加分。
10) 对全部由大一学生组成的参赛队会酌情加分。
11) 发现抄袭现象，直接取消比赛资格。
12) 最终解释权归赛事裁判组所有。

3. 作品提交

3.1. 提交规范

作品请以附件形式发送至大赛官方邮箱: seedcup@dian.org.cn。邮件主题命名方式 :**初赛提交-队名。**要求将完整的工程和工程文档打包提交，压缩包命名规则为：**初赛提交-队名.zip**，提交的目录格式如下：

```
─[初赛提交-队名]
├─Src
├─Bin
├─Doc
└─报名表
```

目录说明如下：Src 文件夹放置源代码及工程文件，Bin 文件夹放置最后生成的可执行文件，Doc 文件夹放置说明文档。这三个文件夹的同级目录下放置报名表，请不要随意修改报名表。

说明文档中需要包含：

1) 程序的编译运行环境说明，包括使用的编译器类型及版本。
2) 程序的设计结构及各模块的功能说明。
3) 对题目功能要求的完成情况，可以列举自己程序中觉得设计良好的部分，并说明为什么好。
4) 你认为必要的附加信息，以方便评委了解你的想法。

3.2. 提交时间

比赛终止时间为 2014/10/26 晚上 10:00，请各位选手注意提交时间（以邮件发送时间为准），我们不接受晚于终止时间提交的作品。

4. 评分说明

我们会依据选手提交上的文档和代码编译可执行文件，并将其作为主要评分标准，选手提交的可执行文件仅作为参考。另外，请特别注意输出的格式（参见 *5.详细需求*），**输出格式错误将直接被认为结果无效。**

项目	分值	评分标准
文档	10’	逻辑和结构清晰，描述和图例详细，不得超过 10 页。
代码规范	10’	编码的规范程度和代码的设计结构，要求核心代码都能给出注释(很重要)。
功能点	80’	功能点的实现情况，依据下面的详细需求包括： 正则中的元字符和反义元字符的匹配(20) 能够匹配重复的字符(20) 支持正则表达式的分支匹配(10) 支持正则表达式的分组功能(30)
附加项		参见 *2.基本要求*中的 8,9,10 三项，我们将会视具体情况酌情给分。

5. 详细需求

5.1. 输入输出规范

1. **文本规范**

最终的可执行文件命名为 SeedCup.exe。同目录下存在命名为 regex.txt 的文本，regex.txt 每一行都记录一个正则表达式。同时，还有 text1.txt，text2.txt，text3.txt…等测试文本。

2. **程序 IO 规范**

SeedCup.exe 会读取 regex.txt 的每一行，用第一行的正则表达式去匹配 text1.txt，用第二行的正则表达式去匹配 text2.txt……

我们要求你能够将每一行的正则表达式对应的输出结果，保存到名为 output1.txt，output2.txt，output3.txt……的文本中，**文件的末尾不要带有额外的换行符。**当输出结果为空时，生成一个空文件即可。

5.2. 需求说明

1. 为了便于识别，下面所有给的例子中的空格符，都直接使用*(空格符)*标记出来，所有的 tab 符，

都直接使用(*tab 符*)标记出来。

2. 下面的例子中的输出为 null 时，代表没有得到匹配结果，直接生成一个空的文档即可。
3. 对于给出的所有例子，我们都按照 *5.1.1 文本规范*提供了对应的文本，方便大家理解和测试。

5.3. 元字符和反义元字符的匹配 (20')

你的正则表达式引擎要求能匹配下面这些基本的元字符，以及表示反义的元字符。为了减小各位选手的工作量，我们不必考虑元字符自身的转义。

'.'(点)

功能： 匹配文本中除了换行符外的任意字符

eg：(text1.txt, text2.txt)

输入：a.c 测试文本：abc 输出：abc

输入：.. 测试文本：a\n 输出 :null(没有匹配结果生成一个空文本文档)

\w

功能：匹配文本中字母或数字

eg：(text3.txt, text4.txt)

输入：\w\w\w 测试文本：1a2b3c 输出：1a2

输入：\w\w\w 测试文本：1a_c 输出：null（空文本）

\s

功能：匹配文本中任意的空白符（包括换行符）

eg：(text5.txt, text6.txt)

输入：\w\s\w 测试文本：a(空格符)bc 输出：a(空格符)b

输入：\s\w 测试文本：abc(tab 符)d 输出：(tab 符)d

\d

功能：匹配文本中的数字

eg：(text7.txt, text8.txt)

输入：\d 测试文本：abc1d 输出：1

输入：\d 测试文本：abcd 输出：null

\b

功能：匹配文本单词的开始或者结束

eg：(text9.txt, text10.txt)

输入：\b\w 测试文本：(空格符)abc1d 输出：a

输入：\b\d 测试文本：(空格符)abc1d 输出 :(null)

^

功能：匹配一行的开始

eg：(text11.txt, text12.txt)

输入：^\w 测试文本：(空格符) (空格符)abc1d 输出：null

输入：^\w 测试文本：abc1d 输出：a

$

功能：匹配一行的结束

eg：(text13.txt, text14.txt)

输入：\w$ 测试文本：abc1d 输出：d

输入：\w$ 测试文本：abc1d(空格符) 输出：null

[x]

功能：x 在此处为代表字符，如[a] 表示匹配 a 字符，[qwe]为匹配 qwe 中的任意一个字符，[\d]表示匹配\字符或者 d 字符。特殊的， [a-zA-Z]可以表示匹配所有的英文字母中的任意一个字符

eg：(text15.txt, text16.txt)

输入：[1] 测试文本：abc1d 输出：1

输入：[a][b-c] 测试文本：aac1d 输出：ac

\W

功能：匹配文本中任意不是字母，数字的字符

eg：(text17.txt, text18.txt)

输入：\W 测试文本：abc1d 输出：null

输入：\W\W\W 测试文本：!@#$abc 输出：!@#

\S

功能：匹配文本中任意不是空白符的字符

eg：(text19.txt, text20.txt)

输入：\S\S\S 测试文本：abcde 输出：abc

输入：\w\S\w 测试文本：a(空格符)bc 输出: null

\D

功能：匹配文本中任意非数字的字符

eg：(text21.txt, text22.txt)

输入：\D\D 测试文本：12ab 输出：ab

输入：\D\D\D 测试文本：123a(空格符)b 输出：a(空格符)b

\B

功能：匹配不是单词开头或结束的位置

eg：(text23.txt, text24.txt)

输入：\B\w 测试文本：(空格符)abc1d 输出：b

输入：\B\d 测试文本：(空格符)abc1d 输出：1

[^x]

功能：匹配除了 x 以外的任意字符 (x 在此处为代表字符，如[^a] 表示匹配除了 a 以外的任意字符，[^qwe]为匹配除了 qwe 外的任意字符，[^\d]表示匹配除了\或者 d 之外的字符。特殊的， [^a-zA-Z]可以表示匹配除了所有的英文字母之外的字符)

eg：(text25.txt, text26.txt)

输入：[^1] 测试文本：abc1d 输出：a

输入：[^a][^a] 测试文本：aac1d 输出：c1

5.4. 正则表达式重复匹配(20’)

正则表达式中有一些符号，是用来表示对前一个字符或分组的重复，例如：
\w+ 匹配一个由数字或字符组成的字符串
\d{7} 匹配 7 位的电话号码
要求你的正则表达式解析引擎能够支持以下字符重复功能：
* 重复零次或多次
+ 重复一次或多次
? 重复零次或一次
{n} 重复 n 次
{n,} 重复 n 次或更多次
{n, m} 重复 n 次到 m 次
而且你的正则引擎必须是贪婪的，匹配即尽可能多的内容。关于贪婪原则，在下面的例子中有所体现。

eg：(text27.txt, text28.txt)
输入： .*
测试文本：aa test1 bb test2 cc
输出： test1 bb test2
解释：由于元字符.可以表示除了换行符外的任意字符，且正则引擎是贪婪的，所以会尽可能多地匹配(限制 1024)。如果是非贪婪的，则会输出： test1

输入：ap*
测试文本：there is an app for apple.
输出：a

5.5. 正则表达式的分支匹配（10’)

解决某些问题时， 我们需要用到正则表达式的分支匹配。 使用了分支匹配的正则表达式将会规定几种规则， 如果满足了其中的任意一种， 都应该当成匹配。 具体方法是使用 | 把不同的正则规则隔开。

eg：(text29.txt)
输入：0\d{2}-\d{8} | 0\d{3}-\d{7}
测试文本：0123443010-12345678sdsdf1230376-30376-2233445
输出：010-12345678
解释：该正则的意思是匹配两种不同的以连字号分割的电话号码。同时，遵循后面提到的贪婪原则，尽可能多地匹配。

5.6. 正则表达式中的分组（30’)

对于正则表达式中的分组， 我们仅要求能实现后项引用。

使用小括号指定一个子表达式后，匹配这个子表达式的文本(也就是此分组捕获的内容)可以在表达式或其它程序中作进一步的处理。默认情况下，每个分组会自动拥有一个组号，规则是：从左向右，以分组的左括号为标志，第一个出现的分组的组号为 1，第二个为 2，以此类推。
后向引用用于重复搜索前面某个分组匹配的文本。

eg：(text30.txt, text31.txt, text32.txt, text33.txt)
输入：(\d+)(\s+\1)
测试文本：abc123(空格符) (空格符) (空格符)123
输出：123(空格符) (空格符) (空格符)123

输入：(\d+)(\s+\1)
测试文本：abc123(空格符) (空格符) (空格符)12
输出：null(空文本)

输入：(\d+)(\s+)(\1\2)
测试文本：123(空格符) (空格符) (空格符)123(空格符) (空格符) (空格符) 456(空格符) 456
输出："123(空格符) (空格符) (空格符)123(空格符) (空格符) (空格符)

输入：((\d+)(\s+)(\d+))\s+(\4\2)
测试文本：abc23(空格符) (空格符)456(空格符)45623cdfad23(空格符)dfe
输出：23(空格符) (空格符)456(空格符)45623
解释 :从左向右 ,第二个出现的左括号包括的表达式为\d+ ,第四个出现的左括号包括的表达式为\d+，所以\2 对应前面第一个\d+所匹配的文本，\4 对应前面第二个\d+所匹配的文本。

5.7. 测试及评分说明

1. 对于每个功能点，我们会设置不同难度的测试用例，根据程序通过的测试用例给分
2. 对于附加项，会根据程序完成的程度给分。

6. 参考

1. 了解正则表达式： http://en.wikipedia.org/wiki/Regular_expression
2. 种子杯网站：http://dian.hust.edu.cn/seedpk/?cat=12