**江苏科技大学**

**实 验 报 告**

课 程： 信息内容安全与系统安全测试

学 院： 计算机学院

学 号： 182210710119

姓 名： 陈四贵

班 级： 1822107101

指导老师： 刘噶琼

目 录

[实验一 网络信息内容识别 1](#_Toc74935286)

[一、实验目的 1](#_Toc74935287)

[二、实验环境 1](#_Toc74935288)

[三、实验内容 1](#_Toc74935289)

[四、实验原理 1](#_Toc74935290)

[五、实验步骤暨实验记录 1](#_Toc74935291)

[六、实验分析总结 5](#_Toc74935292)

[七、实验源代码 6](#_Toc74935293)

[实验二 信息隐藏 8](#_Toc74935294)

[一、实验目的 8](#_Toc74935295)

[二、实验环境 8](#_Toc74935296)

[三、实验内容 8](#_Toc74935297)

[四、实验原理 8](#_Toc74935298)

[五、实验步骤 8](#_Toc74935299)

[六、实验截图 9](#_Toc74935300)

[七、实验分析总结 10](#_Toc74935301)

[实验三 黑盒测试技术 11](#_Toc74935302)

[一、实验目的 11](#_Toc74935303)

[二、实验环境 11](#_Toc74935304)

[三、实验内容 11](#_Toc74935305)

[四、实验原理暨实验步骤 11](#_Toc74935306)

[五、实验截图 13](#_Toc74935307)

[六、实验分析总结 14](#_Toc74935308)

[七、实验代码 14](#_Toc74935309)

[实验四 白盒测试技术 16](#_Toc74935310)

[一、实验目的 16](#_Toc74935311)

[二、实验环境 16](#_Toc74935312)

[三、实验内容 16](#_Toc74935313)

[四、实验原理暨实验步骤 16](#_Toc74935314)

[五、实验截图 17](#_Toc74935315)

[六、实验分析总结 18](#_Toc74935316)

[七、实验代码 18](#_Toc74935317)

# 实验一 网络信息内容识别

## 一、实验目的

通过本实验掌握网络信息的预处理，以及相应的数据挖掘算法。

## 二、实验环境

Windows PC机一台、pycharm、python 3.8环境、SPSS、Excel 2016。

## 三、实验内容

1、编写爬虫程序或利用爬虫工具获取网络数据（例如新冠疫情相关），并对爬取的数据进行预处理。注意，只做爬取，勿做攻击。

2、爬取网址：

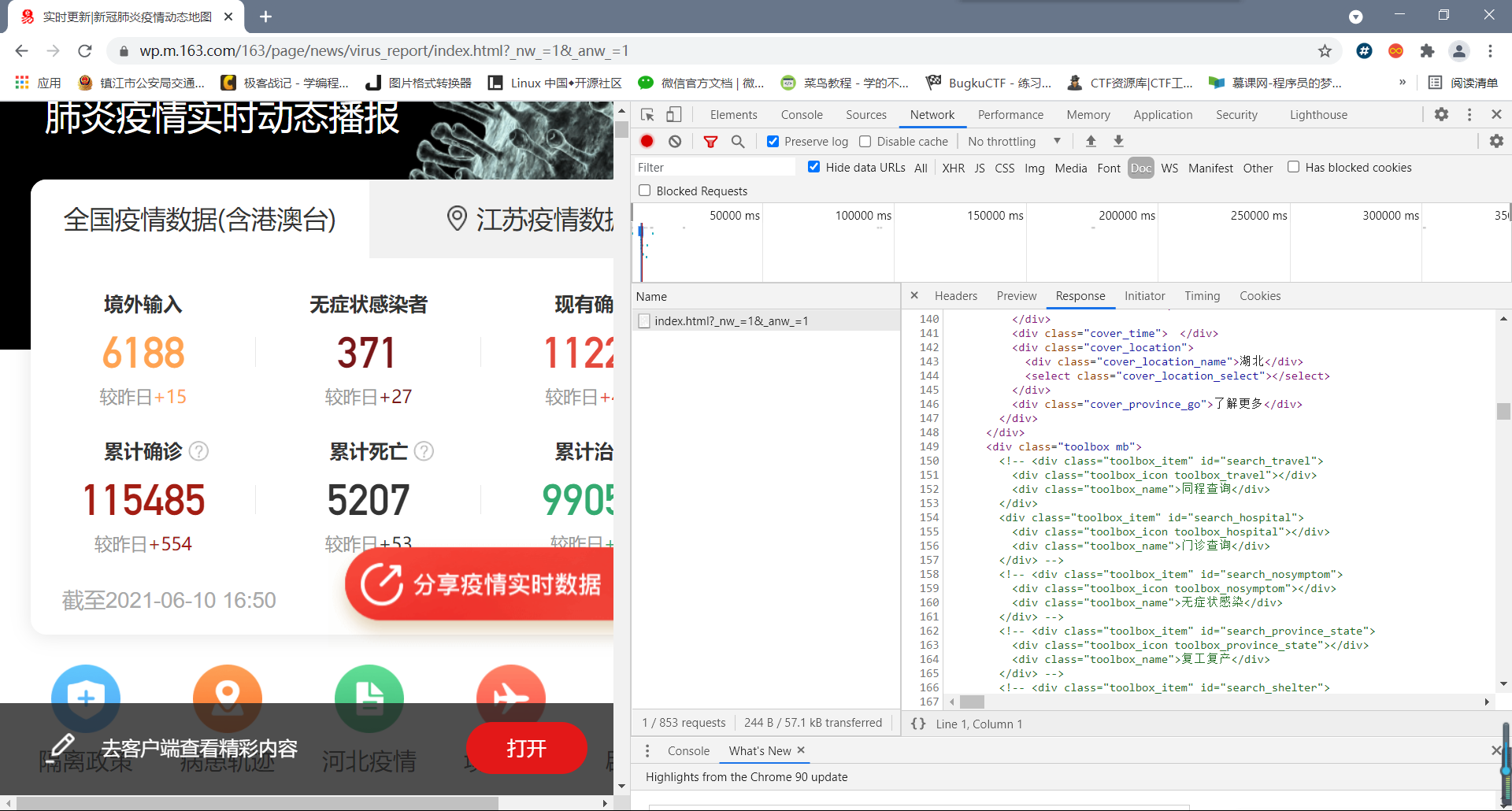
https://wp.m.163.com/163/page/news/virus\_report/index.html?\_nw\_=1&\_anw\_=1

3、利用相关工具从预处理后的数据中挖掘所需相关信息。

## 四、实验原理

通过构造浏览器正常访问网站时的数据包，可以得到响应数据包。根据此数据包，我们可以利用相关工具进行数据挖掘，发现我们想要的信息。信息获取部分可使用爬虫实现，爬虫可以使用python中的requests库等实现。

## 五、实验步骤暨实验记录

1、打开目标网站，利用Chrome自带的Network工具捕获数据包。

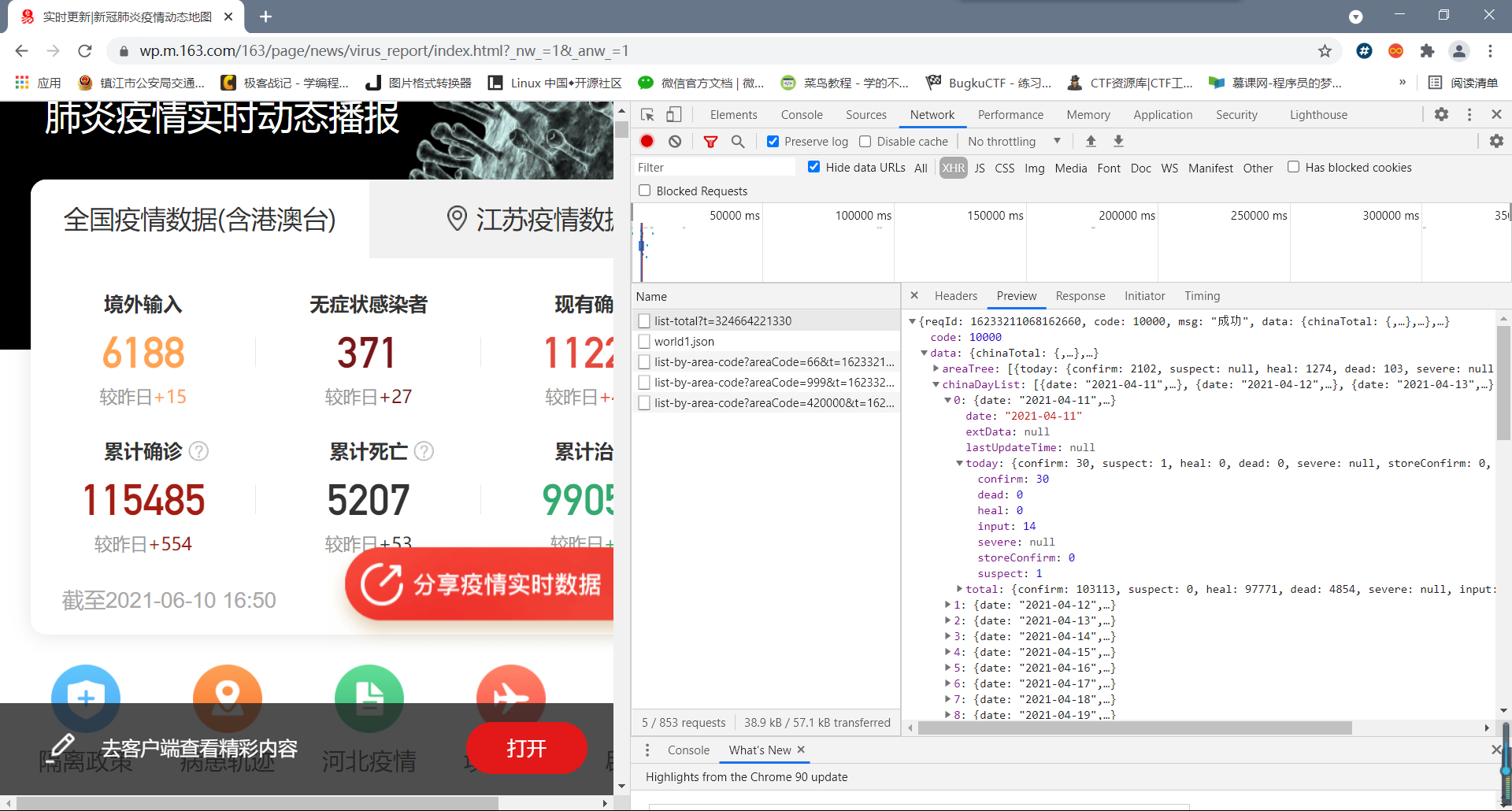
图1-1 捕获数据包

图1-2 捕获数据包

分析后发现，该网页使用了AJAX技术，是一个动态网页。它的数据与页面基本式样是分开的，html页面只是一个空壳，数据通过AJAX技术异步传输。因此，我们若要得到想要的数据部分，需要重点关注XHR请求部分。

2、确定所需信息，比对特征以锁定目标数据包。

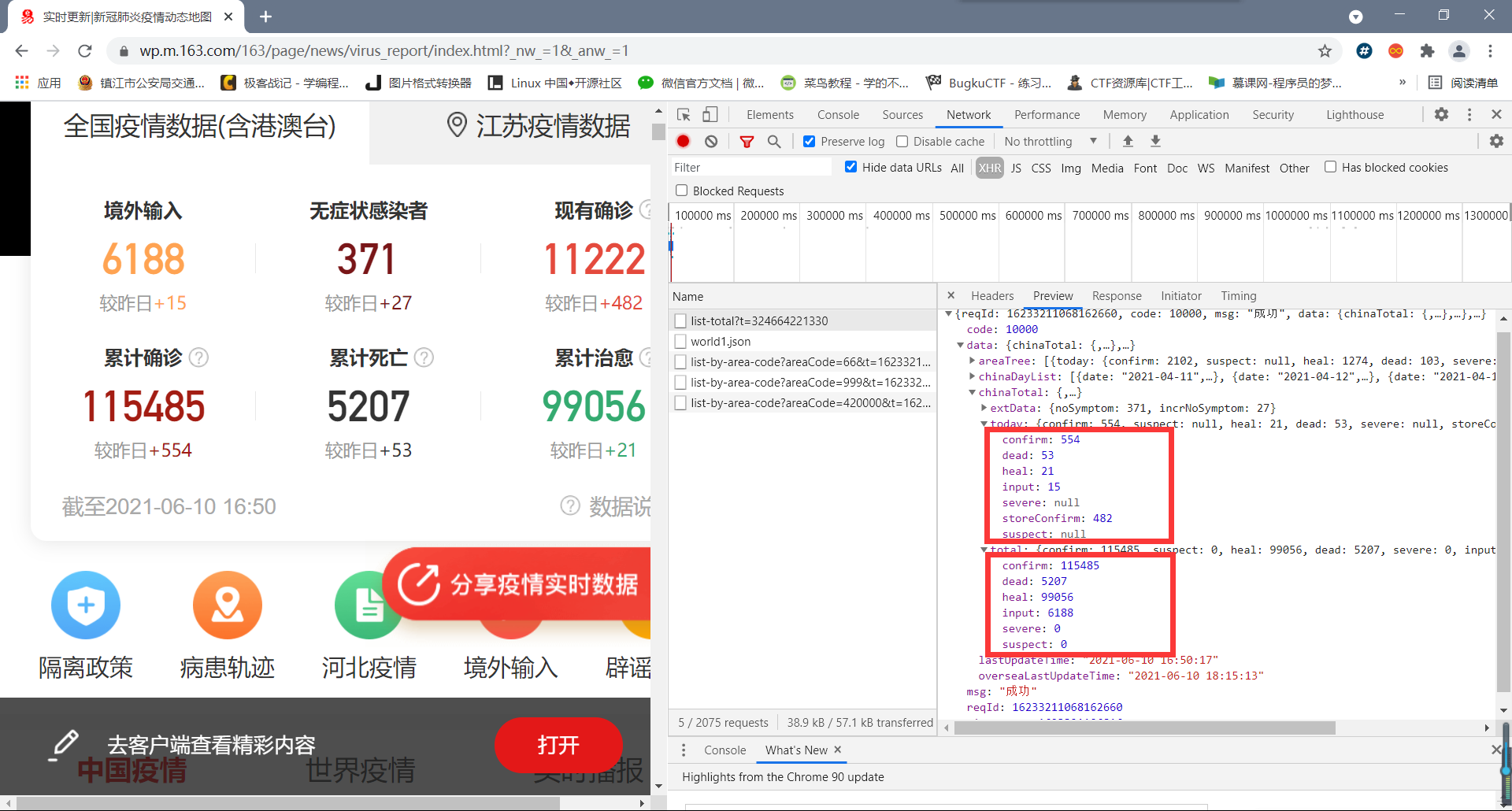
假定我们需要总体情况数据（即当前境外输入、无症状感染者、现有确诊等信息及其当日新增数），则根据数据比对，迅速确定“list-total”数据包为我们所需要的。

图2-1 目标数据包内容

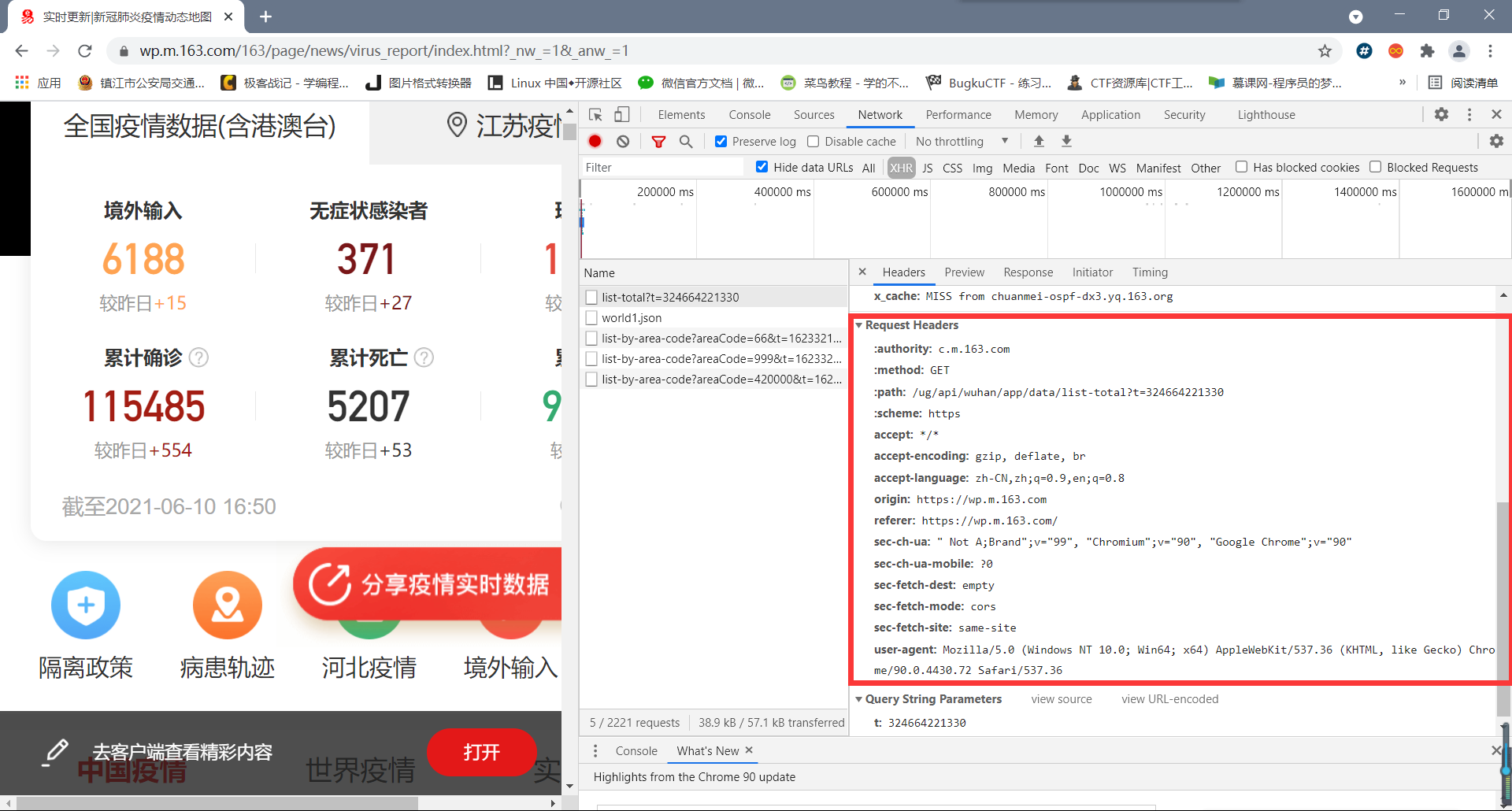
3、根据数据包报头内容，编写请求头以伪装浏览器，达到欺骗服务器的目的。

图3-1 目标数据包报头信息

由此编写请求头为：

1. headers1 = {
2. 'accept': '\*/\*',
3. 'accept-encoding': 'gzip, deflate, br',
4. 'accept-language': 'zh-CN,zh;q=0.9,en;q=0.8',
5. 'origin': 'https://wp.m.163.com',
6. 'referer': 'https://wp.m.163.com/',
7. 'sec-ch-ua': '" Not A;Brand";v="99", "Chromium";v="90", "Google Chrome";v="90"',
8. 'sec-ch-ua-mobile': '?0',
9. 'sec-fetch-dest': 'empty',
10. 'sec-fetch-mode': 'cors',
11. 'sec-fetch-site': 'same-site',
12. 'user-agent': 'Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/90.0.4430.72 Safari/537.36'
13. }

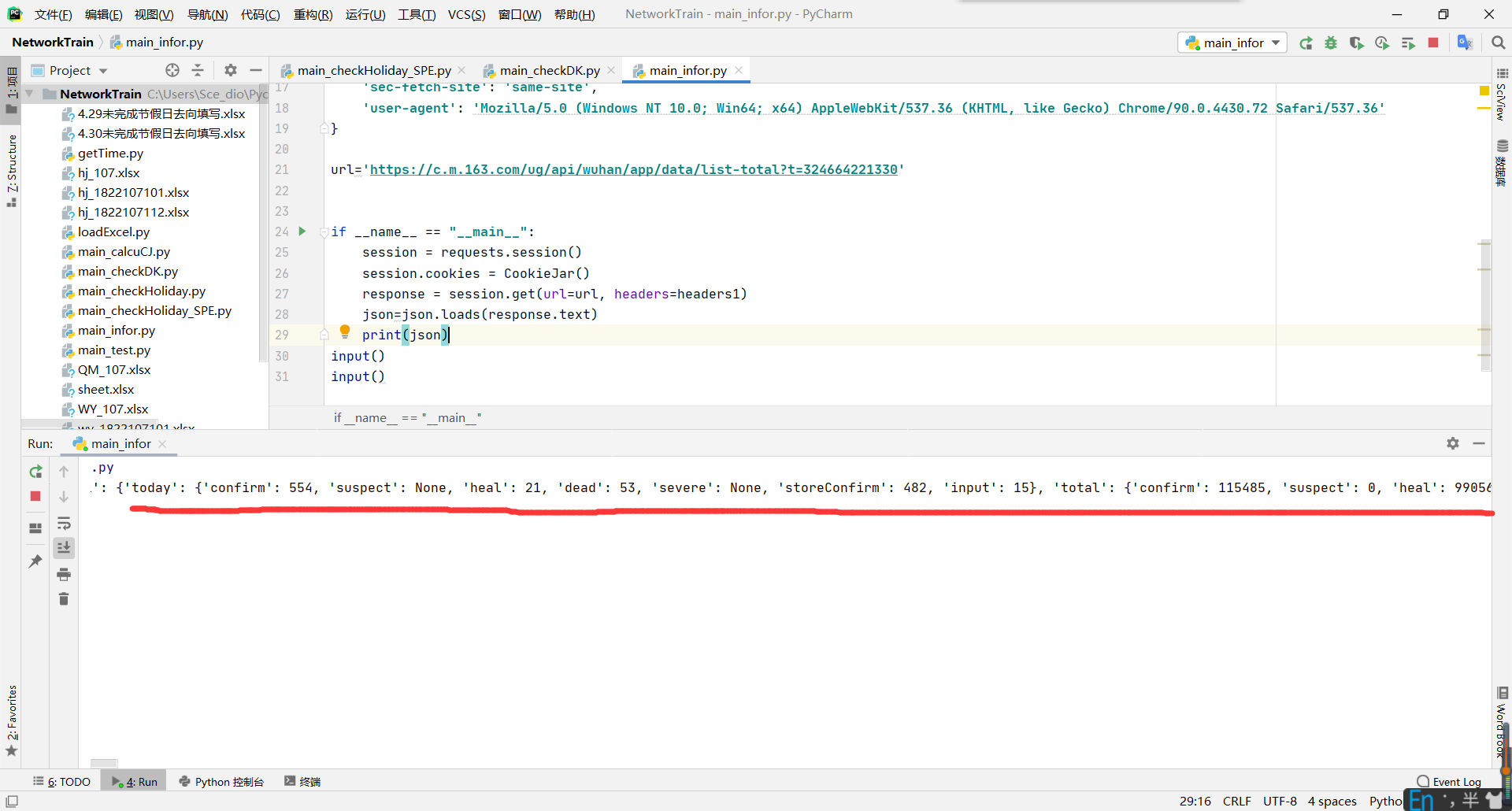
使用json加载后，比较获取的报文内容与我们需要的数据，发现与我们想要的相符。获取成功。

图3-2 获取到的报文内容打印

4、处理json数据，将其保存到excel表格中，以便后续分析。

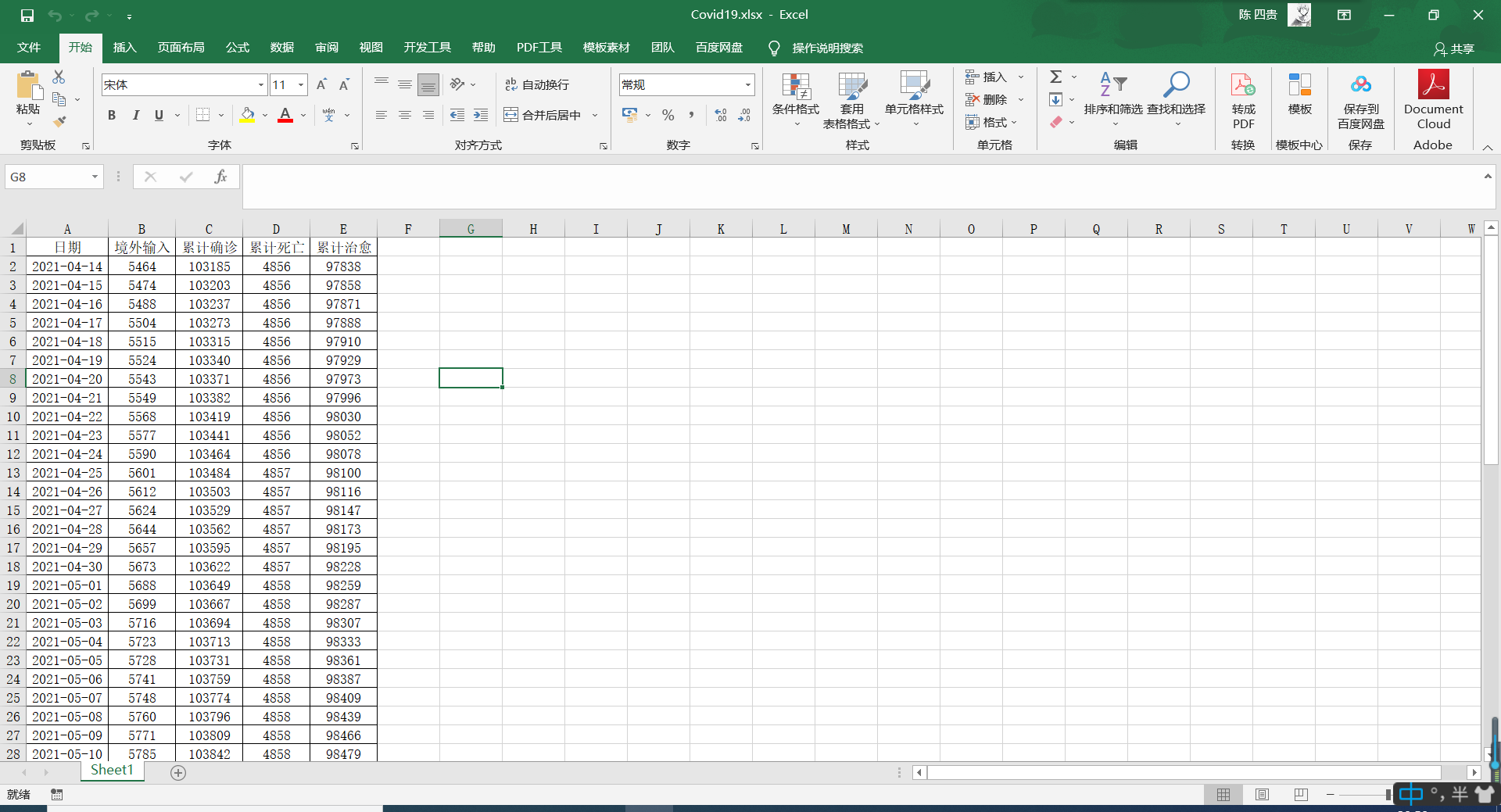
1. f = Workbook()
2. sheet = f.active
3. sheet.title = "Sheet1"
4. row0 = ["日期", "境外输入", "累计确诊", "累计死亡", "累计治愈"]
5. sheet["A1"].value = row0[0]
6. sheet["B1"].value = row0[1]
7. sheet["C1"].value = row0[2]
8. sheet["D1"].value = row0[3]
9. sheet["E1"].value = row0[4]
11. counter = 2
12. **for** i **in** range(len(json)-1):
13. json\_1=json[i]
14. sheet["A" + str(counter)].value = json\_1['date']                         #日期
15. sheet["B" + str(counter)].value = json\_1['total']['input']               #境外输入
16. sheet["C" + str(counter)].value = json\_1['total']['confirm']             #累计确诊
17. sheet["D" + str(counter)].value = json\_1['total']['dead']
18. sheet["E" + str(counter)].value = json\_1['total']['heal']
19. counter = counter + 1
20. f.save("Covid19.xlsx")

图4-1 数据预处理后的excel表格

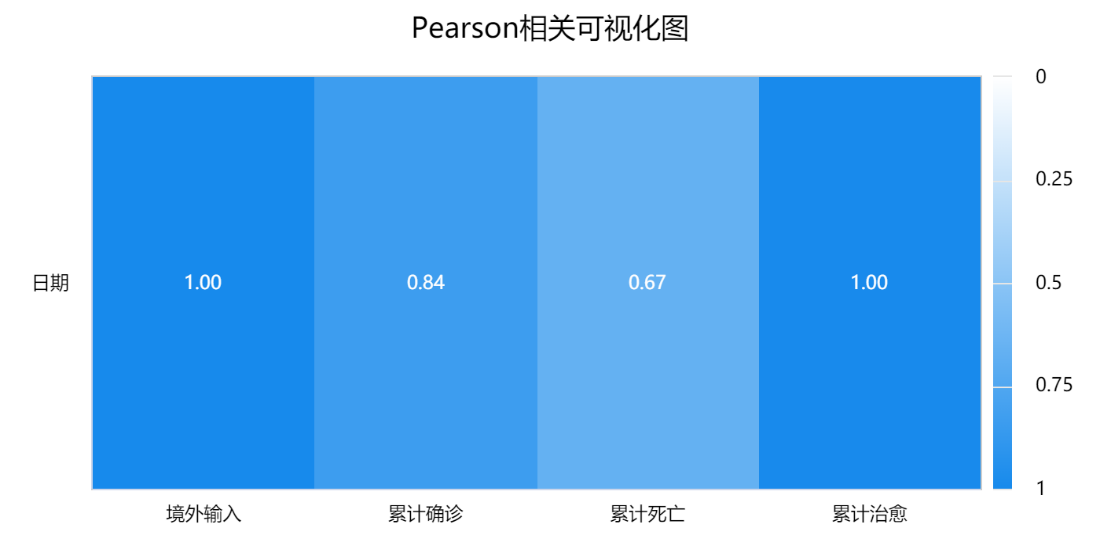
5、对相关数据进行数据挖掘，找出所需信息。

由于我们预处理后的信息是新冠疫情下各项数据随时间的变化情况，因而我们应主要研究数据的变化趋势，根据数据挖掘判断数据的变化趋势、相关性等。

首先可给出各项数据变化趋势折线图，如图5-1所示。

由图中数据可以看出，发生疫情以来，大部分病例是本土病例。在疫情防控常态化下的如今，国内疫情形势已基本得到控制，但仍存在部分新增本土病例和境外输入的问题。此外，随着我国医疗压力的逐步缓解，对新冠肺炎患者的救助能力也在上升，康复率有显著提高。种种数据告诉我们，即使疫苗已经普及，我们仍然不能放松警惕，仍要保持疫情防控常态化，坚决把疫情的势头扼杀在摇篮中，这样才能很好地控制疫情。

然后通过相关分析，我们可以用相关数据科学地描述变化趋势。



从上表可知，利用相关分析去研究境外输入、累计确诊、累计死亡、累计治愈分别和日期共1项之间的相关关系，使用Pearson相关系数去表示相关关系的强弱情况。具体分析可知：

境外输入与日期共1项之间全部均呈现出显著性，相关系数值分别是0.999，并且相关系数值均大于0，意味着境外输入与日期共1项之间有着正相关关系，这可能意味着国外的疫情愈演愈烈。累计确诊与日期共1项之间全部均呈现出显著性，相关系数值分别是0.837，并且相关系数值均大于0，意味着累计确诊与日期共1项之间有着正相关关系。累计死亡与日期共1项之间全部均呈现出显著性，相关系数值分别是0.668，并且相关系数值均大于0，意味着累计死亡与日期共1项之间有着正相关关系。累计治愈与日期共1项之间全部均呈现出显著性，相关系数值分别是0.997，并且相关系数值均大于0，意味着累计治愈与日期共1项之间有着正相关关系。

## 六、实验分析总结

1、数据获取：爬虫是很有效的获取数据的一种方式，但若应用不当，也可能违反法律。在实际爬取的过程中，应注意结合实际分析数据包。比如，可充分利用Chrome Network工具，比对我们所需要的数据，找到数据包。此外，有的网页是静态网页，所有的信息都在html中体现，但有的网页使用了AJAX技术，其网页和数据异步加载。对于这样的网站，我们应着重关注XHR请求，因为我们想要的数据通过此种方式呈现。

2、数据挖掘：数据挖掘是指从大量的数据中通过算法搜索隐藏于其中信息的过程，是人工智能和数据库领域研究的热点问题。在这样的大数据时代，我们应当掌握数据挖掘算法。

## 七、实验源代码

1. **import** requests
2. **import** json
3. **from** http.cookiejar **import** CookieJar
4. **from** openpyxl **import** Workbook
6. headers1 = {
7. 'accept': '\*/\*',
8. 'accept-encoding': 'gzip, deflate, br',
9. 'accept-language': 'zh-CN,zh;q=0.9,en;q=0.8',
10. 'origin': 'https://wp.m.163.com',
11. 'referer': 'https://wp.m.163.com/',
12. 'sec-ch-ua': '" Not A;Brand";v="99", "Chromium";v="90", "Google Chrome";v="90"',
13. 'sec-ch-ua-mobile': '?0',
14. 'sec-fetch-dest': 'empty',
15. 'sec-fetch-mode': 'cors',
16. 'sec-fetch-site': 'same-site',
17. 'user-agent': 'Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/90.0.4430.72 Safari/537.36'
18. }
20. url = 'https://c.m.163.com/ug/api/wuhan/app/data/list-total?t=324664221330'
22. **if** \_\_name\_\_ == "\_\_main\_\_":
23. session = requests.session()
24. session.cookies = CookieJar()
25. response = session.get(url=url, headers=headers1)
26. INfor = json.loads(response.text)
27. json = INfor['data']['chinaDayList']
28. **print**(json)
30. f = Workbook()
31. sheet = f.active
32. sheet.title = "Sheet1"
33. row0 = ["日期", "境外输入", "累计确诊", "累计死亡", "累计治愈"]
34. sheet["A1"].value = row0[0]
35. sheet["B1"].value = row0[1]
36. sheet["C1"].value = row0[2]
37. sheet["D1"].value = row0[3]
38. sheet["E1"].value = row0[4]
39. counter = 2
40. **for** i **in** range(len(json)-1):
41. json\_1=json[i]
42. sheet["A" + str(counter)].value = json\_1['date']                         #日期
43. sheet["B" + str(counter)].value = json\_1['total']['input']               #境外输入
44. sheet["C" + str(counter)].value = json\_1['total']['confirm']             #累计确诊
45. sheet["D" + str(counter)].value = json\_1['total']['dead']
46. sheet["E" + str(counter)].value = json\_1['total']['heal']
47. counter = counter + 1
48. f.save("Covid19.xlsx")
49. **print**("检索完毕，请查阅！")
50. input()

# 实验二 信息隐藏

## 一、实验目的

通过本实验，能使用VS2019的C#进行基本的图像操作，能够对BMP图像实施LSB隐写。掌握W-SVD算法，能熟练对图像添加和检测该类水印。

## 二、实验环境

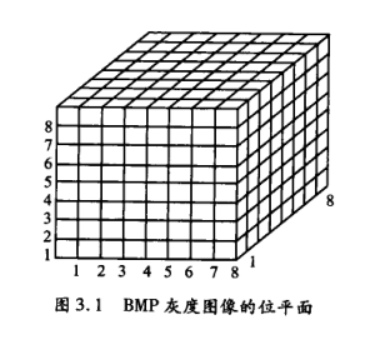
Windows PC机一台、matab R2018a

## 三、实验内容

使用C#或Matlab实现LSB隐写算法实验。

## 四、实验原理

任何多媒体信息在数字化时都会产生物理随机噪声,而人的感观系统对这些随机噪声不敏感。替换技术就是利用这个原理,通过使用秘密信息比特替换随机噪声,从而实现信息隐藏目的。

BMP灰度图像的位平面如图3.1所示,每个像素值为8比特二进制值,表示该点亮度。

不同位平面对视觉影响不同。图像高位平面对图像感官质量起主要作用,去除图像最低几个位平面并不会造成画面质量的明显下降。利用这个原理可用秘密信息(或称水印信息)替代载体图像低位平面以实现信息嵌入。

本算法选用最低位平面来嵌入秘密信息。最低位平面对图像的视觉效果影响最轻微,但很容易受噪声影响和攻击,可采用冗余嵌入的方式来增强稳健性加以解决，即在一个区域(多个像素)中嵌入相同的信息,提取时根据该区域中的所有像素判断。

## 五、实验步骤

1. % 水印嵌入
2. I=imread('lena512.bmp');                       %载入载体图像
3. I=im2double(I);
4. %I=rgb2gray(I);                              %转为灰度图
5. [LL,LH,HL,HH]=dwt2(I,'haar');               %载体图像dwt变换
6. [U,S,V]=svd(LL);                            %低频分量SVD分解
7. W=imread('780.jpg');                    %载入水印图像
8. W=im2double(W);
9. W=rgb2gray(W);                              %转为灰度图
10. [Uw,Sw,Vw]=svd(W);                          %水印图SVD分解
11. af=0.2;                                     %嵌入量
12. S2=S+af\*Sw;                                 %水印图的奇异值矩阵Sw以af嵌入量加到载体图像LL子代的奇异值矩阵S上
13. LL2=U\*S2\*V';                                %嵌入后新的低频分量
14. IW=idwt2(LL2,LH,HL,HH,'haar');              %dwt逆变换得到含水印图
15. imwrite(IW,'watermarked.bmp');
16. figure,
17. subplot(221);imshow(I);title('载体图像');
18. subplot(222);imshow(W);title('水印图像');
19. subplot(223);imshow(IW);title('含水印图');
20. % 水印提取
21. J=imread('watermarked.bmp');                %读取含水印图
22. J=im2double(J);
23. [LL3,LH3,HL3,HH3]=dwt2(J,'haar');           %含水印图dwt变换
24. [U3,S3,V3]=svd(LL3);                        %低频分量SVD分解
25. S4=(S3-S)/af;                               %得到提取水印图的奇异值矩阵
26. W2=Uw\*S4\*Vw';                               %SVD逆变换得到提取水印图
27. subplot(224);imshow(W2);title('提取水印图');

## 六、实验截图

## 七、实验分析总结

分析：

本次实验我们实现了LSB图像隐藏算法，其方法是用嵌入的秘密信息取代载体图像的最低比特位，原来图像的7个高位平面与代表秘密信息的最低位平面组成含隐蔽图像的新图像。虽然LSB隐写在隐藏大量信息的情况下依然保持良好的视觉隐蔽性，但使用有效的统计分析工具仍可判断一幅载体图像中是否含有秘密信息。

比如，可利用卡方分析、RS分析等手段判断一幅载体图像中是否含有秘密信息。即使这些分析算法也有其本身的不足，但毕竟是可以尝试的方法。

总得来说，LSB算法有较大的信息隐藏量，实现起来比较简单。但采用此方法生成的水印相对而言比较脆弱，鲁棒性差，难以适应实际应用中复杂的使用环境。此外，当嵌入消息较大时，所花时间较长，且只能处理简单的流格式的文件，有很大的束缚。

总结：

1、诸如LSB的图像隐藏算法生成的水印较脆弱，鲁棒性差。若在战争期间发生信息战时，我们可以对国家范围内的无线电等可以截获到的信息进行简单的处理，这样敌方（间谍）等就难以把嵌入秘密信息的载体完好无损地传递出去，敌对组织也很难提取完整的信息。此种举措虽然比较麻烦，但有利于我国的国防建设。缺点是伤敌一千，自损八百。

2、在图像信息隐藏中，很多时候都是利用人体视觉系统的不敏感设计嵌入算法。我们在实际应用中应当注重这一方面，不要没有根据地设计嵌入算法，因为那不科学，没有经过系统得考量。

# 实验三 黑盒测试技术

## 一、实验目的

学习和掌握黑盒测试基本技术，并能够应用等价类划分和边界值分析等方法设计测试用例；设计测试用例、执行测试和编写软件问题报告。

## 二、实验环境

Windows PC机一台、visual studio 2017

## 三、实验内容

输入三个整数作为三边的边长构成三角形，分别用等价类方法和判定表方法构造测试用例进行测试（注：三角形可能为一般三角形，等腰三角形，等边三角形）。

## 四、实验原理暨实验步骤

1、等价类方法：等价类划分，指的是一种典型的、重要的黑盒测试方法。其就是解决如何选择适当的数据子集来代表整个数据集的问题，通过降低测试的数目去实现“合理的”覆盖，以此发现更多的软件缺陷，统计好数据后由此对软件进行改进升级。我们将可能的数据输入划分等价类：

|  |  |  |
| --- | --- | --- |
|  | 有效类 | 无效类 |
| 是否是三角形 | a>0(1) | a<=0(7) |
| b>0(2) | b<=0(8) |
| c>0(3) | c<=0(9) |
| a+b>c(4) | a+b<=c(10) |
| a+c>b(5) | a+c<=b(11) |
| b+c>a(6) | b+c<=a(12) |
| 是否是等腰三角形 | a=b(13) | a!=b且b!=c且c!=a(16) |
| a=c(14) |
| b=c(15) |
| 是否是等边三角形 | a=b=c(17) | a!=b(18) |
| a!=c(19) |
| b!=c(20) |

由此，我们确定测试用例，并展开测试，结果如下：

|  |  |  |  |
| --- | --- | --- | --- |
| 序号 | 等价类 | 测试用例 | |
| 输入(a,b,c) | 输出 |
| 1 | (1)、(2)、(3)、(4)、(5)、(6)、(16) | 3,4,5 | 此三条边可以构成普通三角形 |
| 2 | (7) | 0,1,2 | 此三条边不能构成三角形 |
| 3 | (8) | 1,0,2 | 此三条边不能构成三角形 |
| 4 | (9) | 1,2,0 | 此三条边不能构成三角形 |
| 5 | (10) | 1,2,3 | 此三条边不能构成三角形 |
| 6 | (11) | 3,1,2 | 此三条边不能构成三角形 |
| 7 | (12) | 1,3,2 | 此三条边不能构成三角形 |
| 8 | (1)、(2)、(3)、(4)、(5)、(6)、（13）、(19) | 3,3,4 | 此三条边可以构成等腰三角形 |
| 9 | (1)、(2)、(3)、(4)、(5)、(6)、（14）、(20) | 4,3,3 | 此三条边可以构成等腰三角形 |
| 10 | (1)、(2)、(3)、(4)、(5)、(6)、（15）、(18) | 3,4,3 | 此三条边可以构成等腰三角形 |
| 11 | (1)、(2)、(3)、(4)、(5)、(6)、（17） | 3,3,3 | 此三条边可以构成等边三角形 |

2、判定表方法：是分析和表达多逻辑条件下执行不同操作的情况的工具，能够将复杂的问题按照各种可能的情况全部列举出来，简明并避免遗漏。其中，条件桩列出了问题的所有条件，通常认为列出的条件的次序无关紧要；动作桩列出了问题规定可能采取的操作，这些操作的排列顺序没有约束；条件项列出了针对条件的取值和在所有可能情况下的真假值；具体项列出了在条件项的各种取值情况下应该采取的动作。建立判定表如下：

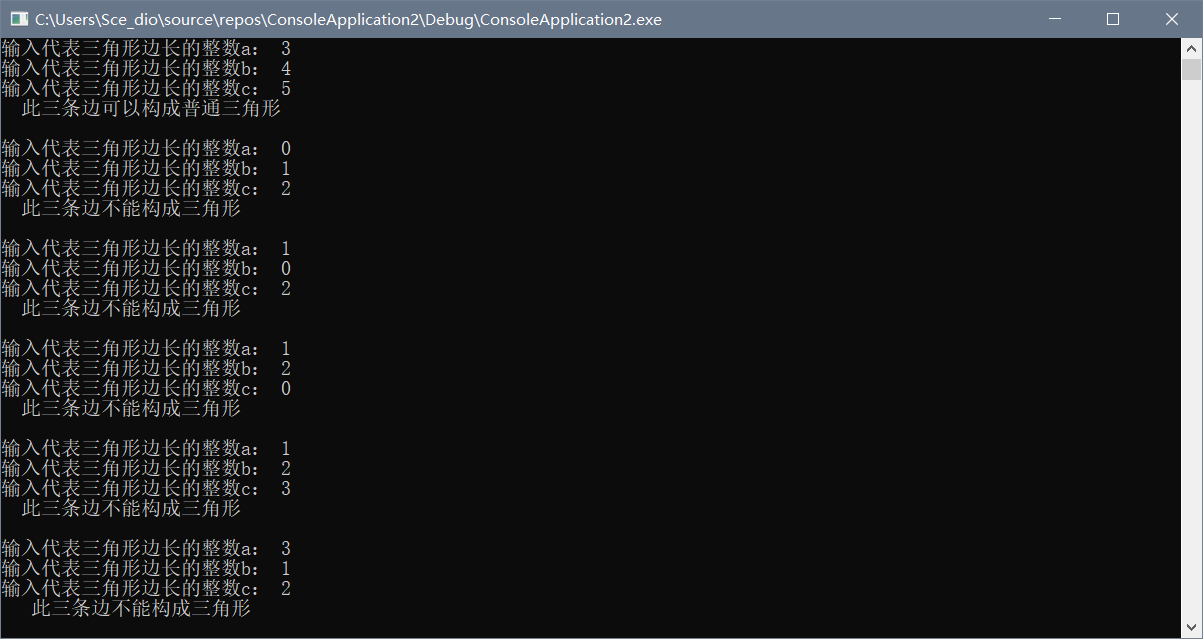
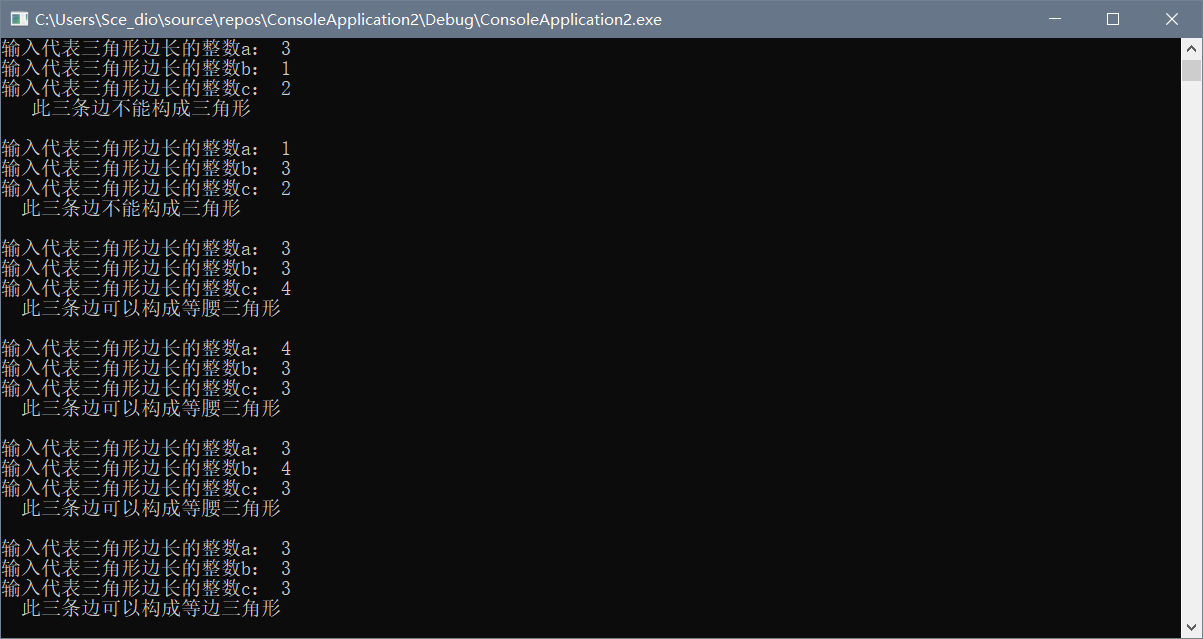
|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 规则 | | 1~8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 |
| 条件桩 | C1：a,b,c能组成三角形 | N | Y | Y | Y | Y | Y | Y | Y | Y |
| C2：a=b | - | N | N | N | N | Y | Y | Y | Y |
| C3：b=c | - | Y | Y | N | N | Y | Y | N | N |
| C4：a=c | - | Y | N | Y | N | Y | N | Y | N |
| 动作桩 | A1：不能构成三角形 | √ |  |  |  |  |  |  |  |  |
| A2：普通三角形 |  |  |  |  | √ |  |  |  |  |
| A3：等腰三角形 |  |  | √ | √ |  |  |  |  | √ |
| A4：等边三角形 |  |  |  |  |  | √ |  |  |  |
| A5：不符合逻辑 |  | √ |  |  |  |  | √ | √ |  |

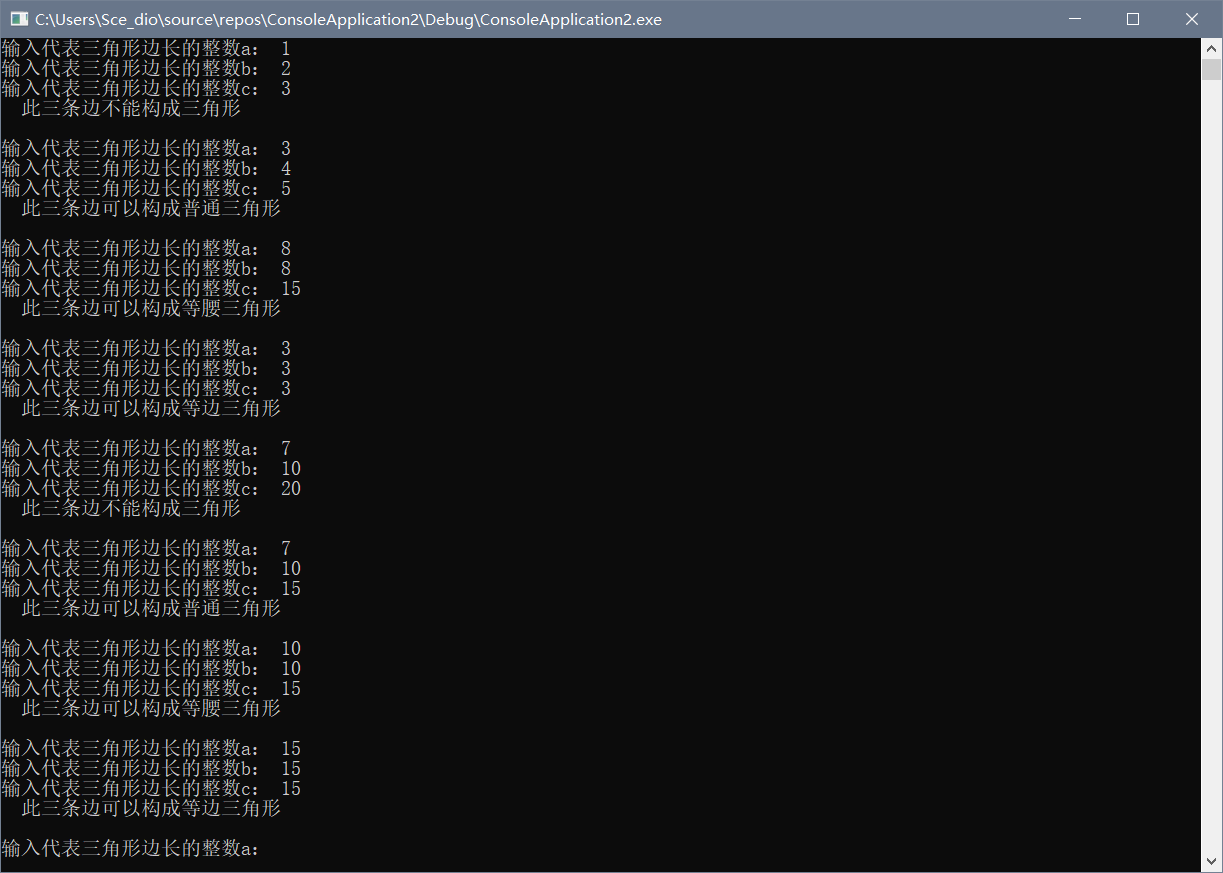
由此，确定测试用例，并开展测试，结果如下：

|  |  |  |  |
| --- | --- | --- | --- |
| 用例ID | 规则 | 输入（a,b,c） | 输出 |
| 1 | 不能组成三角形 | 1,2,3 | 不是三角形 |
| 2 | 能组成三角形 | 3,4,5 | 普通三角形 |
| 3 | a、b、c能组成三角形，且a=b | 8,8,15 | 等腰三角形 |
| 4 | a、b、c三边能组成三角形，且a=b，b=c | 3,3,3 | 等边三角形 |
| 5 |  | 7,10,20 | 不是三角形 |
| 6 |  | 7,10,15 | 普通三角形 |
| 7 |  | 10,10,15 | 等腰三角形 |
| 8 |  | 15,15,15 | 等边三角形 |

由此可以判断，该程序能很好地判断是否能构成三角形，以及是普通三角形还是等腰（边）三角形的问题。其对变量互换位置具有很强的鲁棒性，在一定程度上能适应用户的输入。

## 五、实验截图

1、等价类方法

2、判定表方法

## 六、实验分析总结

黑盒测试是软件测试中的重要一环，相比于按照程序内部结构进行测试的白盒测试，黑盒测试要更加贴切于大众，更加贴切于实际情况。因为我们开发时很难完全预见到用户所有的输入，又或者是网络情况而导致服务器载荷的变化出现的问题，因此，在系统上线前，对其做黑盒测试是很有必要的。

## 七、实验代码

1. #include <iostream>
2. **using** **namespace** std;
3. **int** main()
4. {
5. **do** {
6. **int** a, b, c;
7. cout << "输入代表三角形边长的整数a： ";
8. cin >> a;
9. cout << "输入代表三角形边长的整数b： ";
10. cin >> b;
11. cout << "输入代表三角形边长的整数c： ";
12. cin >> c;
13. **if** (a < b + c)
14. {
15. **if** (b < c + a)
16. {
17. **if** (c < a + b)
18. {
19. **if** (a == b)
20. {
21. **if** (b == c)
22. cout << "  此三条边可以构成等边三角形   ";
23. **else**
24. cout << "  此三条边可以构成等腰三角形   ";
25. }
26. **else**
27. {
28. **if** (a == c)
29. cout << "  此三条边可以构成等腰三角形   ";
30. **else**
31. {
32. **if** (b == c)
33. cout << "  此三条边可以构成等腰三角形   ";
34. **else**
35. cout << "  此三条边可以构成普通三角形   ";
36. }
37. }
38. }
39. **else**
40. cout << "  此三条边不能构成三角形   ";
41. }
42. **else**
43. cout << "  此三条边不能构成三角形   ";
44. }
45. **else**
46. cout << "   此三条边不能构成三角形    ";
47. cout << "\n\n";
48. } **while** (1 > 0);
49. system("pause");
50. }

# 实验四 白盒测试技术

## 一、实验目的

实践白盒测试基本方法（语句覆盖、分支覆盖、条件覆盖、分支/条件覆盖、条件组合覆盖及基本路径测试方法）。

## 二、实验环境

Windows PC机一台、visual studio 2017

## 三、实验内容

定义一个PriorDate函数，PriorDate函数为了获得当前输入日期的前一个日期， 执行如下操作：

如果输入日期day变量值大于1，则把day变量的值减1；

如果输入日期是2~12月份中某月的第一天，则把day变量的值置为前一个月的最后一天，month变量的值减1；

如果输入日期是1月的第一天，则day变量的值置为31，month变量的值置为12，year变量的值减1。

关于最后一天的判断：

如果是有31天的月份(1,3,5,7,8,10,12)，day变量值为31；

如果是有30天的月份(4,6,9,11)，day变量值为30；

如果是有29天的月份(闰年的2月)，day变量值为29；

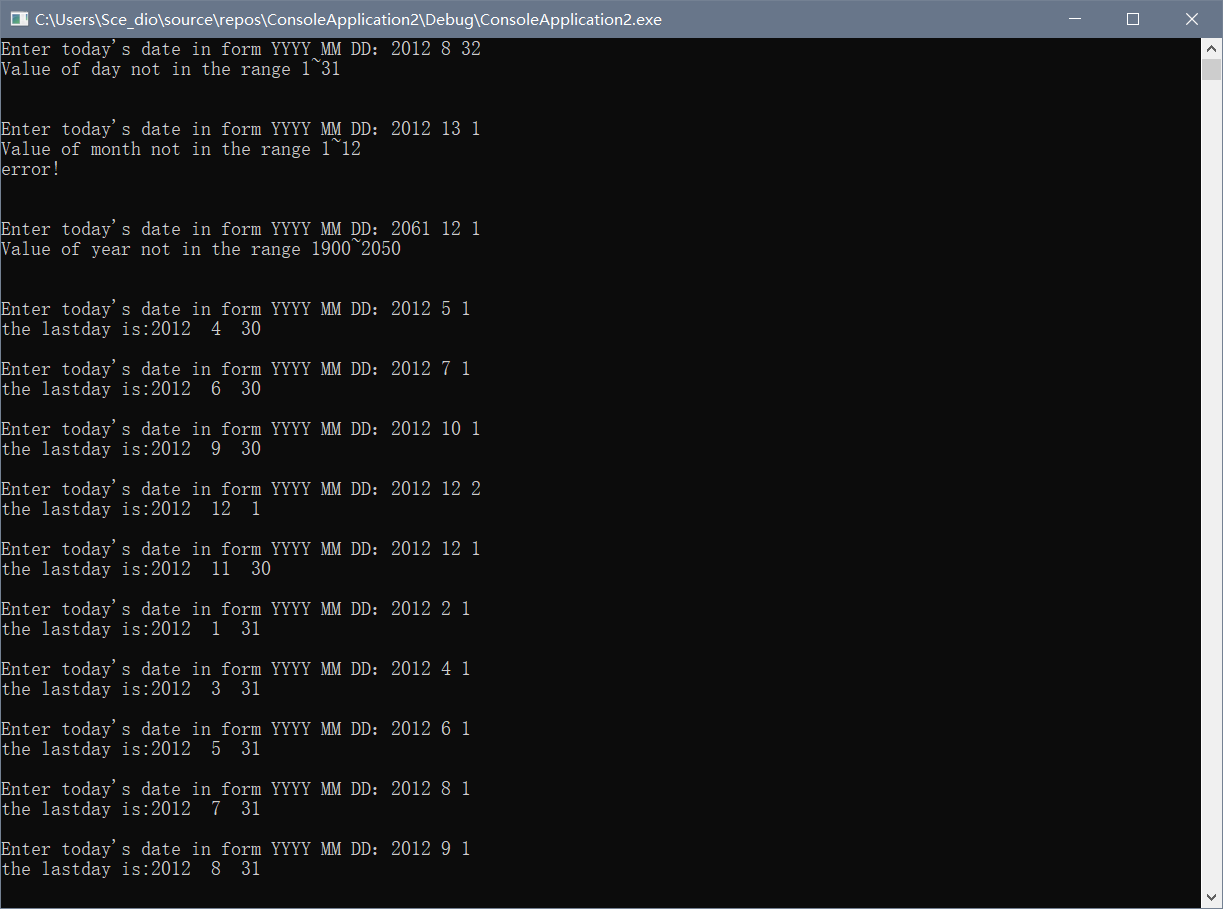
如果是有28天的月份(非闰年的2月)，day变量值为28。

## 四、实验原理暨实验步骤

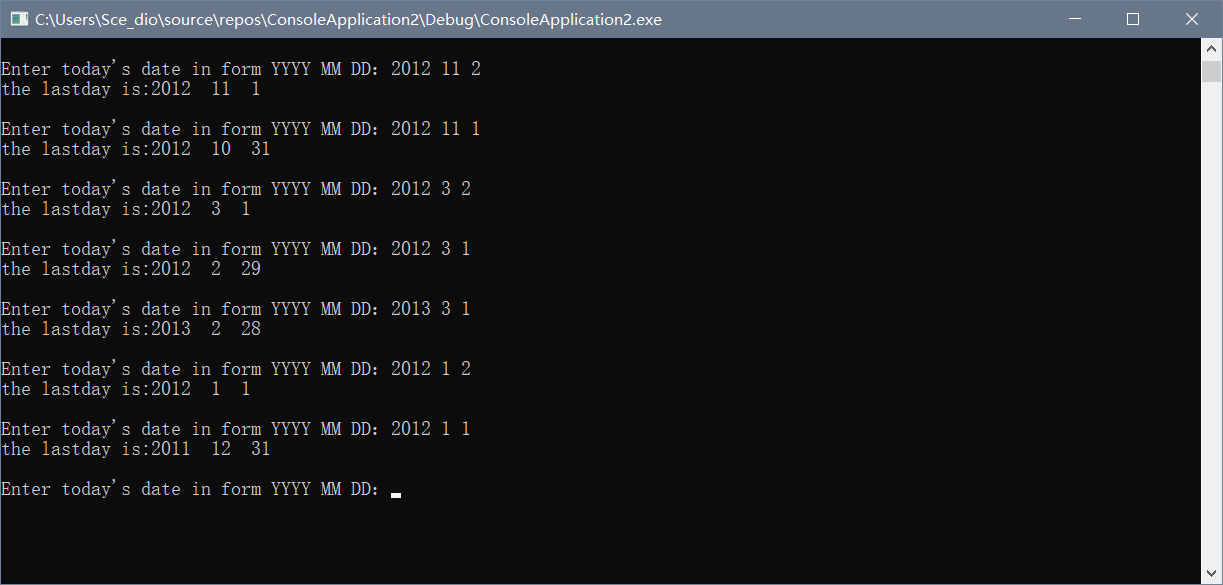
我们使用分支/条件覆盖，程序中每个判定至少有一次为真值，有一次为假值,使得程序中每个分支至少执行一次，由此，我们确定测试用例并开展测试，结果如下：

|  |  |  |
| --- | --- | --- |
| 测试ID | 测试用例 | 测试结果 |
| 1 | 2012 8 32 | Value of day not in the range 1~31 |
| 2 | 2012 13 1 | Value of month not in the range 1~12  error! |
| 3 | 2061 12 1 | Value of year not in the range 1900~2050 |
| 4 | 2012 5 1 | the lastday is:2012 4 30 |
| 5 | 2012 7 1 | the lastday is:2012 6 30 |
| 6 | 2012 10 1 | the lastday is:2012 9 30 |
| 7 | 2012 12 2 | the lastday is:2012 12 1 |
| 8 | 2012 12 1 | the lastday is:2012 11 30 |
| 9 | 2012 2 1 | the lastday is:2012 1 31 |
| 10 | 2012 4 1 | the lastday is:2012 3 31 |
| 11 | 2012 6 1 | the lastday is:2012 5 31 |
| 12 | 2012 8 1 | the lastday is:2012 7 31 |
| 13 | 2012 9 1 | the lastday is:2012 8 31 |
| 14 | 2012 11 2 | the lastday is:2012 11 1 |
| 15 | 2012 11 1 | the lastday is:2012 10 31 |
| 16 | 2012 3 2 | the lastday is:2012 3 1 |
| 17 | 2012 3 1 | the lastday is:2012 2 29 |
| 18 | 2013 3 1 | the lastday is:2013 2 28 |
| 19 | 2012 1 2 | the lastday is:2012 1 1 |
| 20 | 2012 1 1 | the lastday is:2011 12 31 |

由此可以看出，该程序非常优秀地实现了我们应有的功能，程序逻辑严密、功能齐全，能判断闰年、平年并由此判断出2月的天数，能判定某月的具体天数，能判定某日是否为某月的最后一天……总之，在正确的输入下，该程序能实现应有的功能。



## 五、实验截图



## 六、实验分析总结

白盒测试又称结构测试、透明盒测试、逻辑驱动测试或基于代码的测试。白盒测试是一种测试用例设计方法，盒子指的是被测试的软件。在对软件进行白盒测试时，我们应当对程序内部的结构有着比较清晰的了解，然后按照程序本身的结构去测试，得到我们应得的结果。可以说，白盒测试就是检测系统能否按照自身编写时所预想的那样进行。它只能代表系统可以正常运行，而不能代表系统可以经受得住用户的考验。白盒测试并不能代替黑盒测试。

## 七、实验代码

1. #include <iostream>
2. **using** **namespace** std;
3. **int** main()
4. {
5. **do** {
6. **int** lastday, lastmonth, lastyear;
7. **int** day, month, year;
8. **bool** c1 = 1, c2 = 1, c3 = 1;
9. **while** (c1&&c2&&c3)
10. {
11. cout << "Enter today's date in form YYYY MM DD：" ;
12. cin >> year >> month >> day;
13. c1 = (day >= 1) && (day <= 31);
14. c2 = (month >= 1) && (month <= 12);
15. c3 = (year >= 1900) && (year <= 2050);
16. **if** (!c1)
17. cout << "Value of day not in the range 1~31" << "\n";
18. **if** (!c2)
19. cout << "Value of month not in the range 1~12" << "\n";
20. **if** (!c3)
21. cout << "Value of year not in the range 1900~2050" << "\n";
22. **switch** (month)
23. {
24. **case** 5:
25. **case** 7:
26. **case** 10:
27. **case** 12:
28. **if** (day > 1)
29. {
30. lastday = day - 1;
31. lastmonth = month;
32. lastyear = year;
33. }
34. **else**
35. {
36. lastday = 30;
37. lastmonth = month - 1;
38. lastyear = year;
39. }
40. **break**;
41. **case** 2:
42. **case** 4:
43. **case** 6:
44. **case** 8:
45. **case** 9:
46. **case** 11:
47. **if** (day > 1)
48. {
49. lastday = day - 1;
50. lastmonth = month;
51. lastyear = year;
52. }
53. **else**
54. {
55. lastday = 31;
56. lastmonth = month - 1;
57. lastyear = year;
58. }
59. **break**;
60. **case** 3:
61. **if** (day > 1)
62. {
63. lastday = day - 1;
64. lastmonth = month;
65. lastyear = year;
66. }
67. **else**
68. {
69. **if** (year % 4 == 0 && year % 100 != 0 || year % 400 == 0)
70. {
71. lastday = 29;
72. lastmonth = 2;
73. lastyear = year;
74. }
75. **else**
76. {
77. lastday = 28;
78. lastmonth = 2;
79. lastyear = year;
80. }
81. }
82. **break**;
83. **case** 1:
84. **if** (day > 1)
85. {
86. lastday = day - 1;
87. lastmonth = month;
88. lastyear = year;
89. }
90. **else**
91. {
92. lastday = 31;
93. lastmonth = 12;
94. **if** (year != 1900)
95. lastyear = year - 1;
96. **else** cout << "lastyear is not in range" << endl;
97. }
98. **break**;
99. **default**:
100. cout << "error!" << endl;
101. }
102. **if** (c1&&c2&&c3)
103. cout << "the lastday is:" << lastyear << "  " << lastmonth << "  " << lastday << "\n" << endl;
104. }
105. cout << "\n\n";
106. }**while** (1 > 0);
107. **return** 0;
108. }