问题8

1 任务简介

您将使用面向对象的编程（类和继承）来构建一个程序来监视 Internet 上的新闻源。您的程序将过滤新闻，当用户注意到与该用户的兴趣相匹配的新闻故事时发出警报（例如，每当发布与bilibili有关的故事时，用户可能对通知感兴趣）。

Internet上的新闻源，是新闻的聚合体，通常简称RSS（Really Simple Syndication）。例如http://www.chinanews.com/rss/scroll-news.xml，会及时更新当天的即时新闻；其他类型的新闻，可以参考[2]。

我们可以每30分钟去网站上读取RSS的内容。RSS中的内容可能有差异，但多数包括如下的内容：

```
-新闻标题
-新闻链接
-新闻描述
-新闻时间
```

在网页中，使用类似HTML的标签进行描述，例如

```
<item><title>桂林市漓江水库群三座水库开闸排洪</title>
<link>http://www.chinanews.com/sh/2020/06-08/9206720.shtml</link><description>中新网桂林6月8日电(刘建设 欧惠兰)记者从桂林市漓江水库群管理部门获悉，6月8日18时，桂林市漓江水库群斧子口、小溶江、川江三座水库再度开闸排洪，以腾出部分库容防御后续强降雨，为漓江拦洪削峰。</description><pubDate>2020-06-08 20:41:15</pubDate></item>
```

大量的新闻被下载下来之后，一定要先进行过滤，只有满足特定条件的新闻才显示给用户看。过滤时针对每一条新闻，过滤的条件可以包括：

```
-新闻标题:新闻标题中包含特定的关键字，关键字不区分大小写；
-新闻链接：不支持过滤
-新闻描述:新闻描述中包含特定的关键字，关键字不区分大小写；
-新闻时间：在指定时间点之前；或指定时间点之后；
```

例如，新闻标题中包含"上海"，新闻描述中包含"夜市"，新闻时间在"2020/6/6 8:00"之前等

为了得到有效的结果，我们还需要把过滤条件进行组合，即支持

```
-not:把指定的过滤条件取反；
-or:可以把两个指定的条件进行或操作(or)；
-and:可以把两个指定的条件进行与操作(and)；
```

例如，新闻标题中包含"上海"且新闻描述中包含"夜市"。

2 新闻

2.1 新闻的表示

可以定义新闻类，包含属性：title_, link_, description_, pubdate_，以及guid_。引入guid_是为了对新闻进行唯一的标识，类似于身份证的功能。

```
class NewsStory
{
public:
    ...
private:
    std::string guid_;
    std::string title_;
    std::string link_;
    std::string description_;
    std::chrono::system_clock::time_point pub_date_;
};
std::ostream &operator<<(std::ostream &os, const NewsStory &story);
std::istream &operator>>(std::istream &is, NewsStory &story);
```

2.2 新闻的读取

这一部分，使用了SFML的network功能和C++的re功能（Regular Expression, 正则表达式功能）。核心的代码也分为两个部分，分别是网络读取，然后是re处理。

```
NewsStories do_fetch(const std::string& header, const std::string& uri)
{
    // 使用Http读取网络的内容
    sf::Http http{header};
    sf::Http::Request request;
    request.setMethod(sf::Http::Request::Get);
    request.setUri(uri);
    request.setHttpVersion(1, 1); // HTTP 1.1
    sf::Http::Response response = http.sendRequest(request);
    std::string content=response.getBody();

    // 使用re进行处理读取的文件
    NewsStories stories;

    std::regex pattern{"<item><title>([\\s\\S]*?)</title><link>([\\s\\S]*?)
</link><description>([\\s\\S]*?)</description><pubDate>([\\s\\S]*?)</pubDate>
</item>"};
    std::smatch result;
    std::string::const_iterator start = content.begin();
    std::string::const_iterator end = content.end();
    auto count = 1;
    while(std::regex_search(start, end, result, pattern,
std::regex_constants::match_any))
```

```
    {
        std::string title = result[1];
        std::string link = result[2];
        std::string description = result[3];
        std::string pubdate = result[4];
        description = trim(description);

        NewsStory story;
        story.set(title, link, description, pubdate);
        std::cout << count << ": "<< story << "\n";
        stories.push_back(story);

        start = result[0].second;
        ++count;
    }

    return stories;
}
```

仅需调用

```
do_fetch("http://www.chinanews.com/", "rss/scroll-news.xml");
```

就可以读取上面网址下面的内容。

3 过滤

3.1 对于过滤的考虑

用户的兴趣点各不相同，因此最好把用户的兴趣通过一个文件描述出来。例如

```
// trigger file

// title trigger named t1
t1,TITLE,election
// description trigger named t2
t2,TITLE,中国
// description trigger named t3
t3,TITLE,暴雨
// after trigger named t4
t4,AFTER,2020-06-06 17:00:10
// composite trigger named t5
t5,AND,t2,t3
// composite trigger named t6
t6,AND,t1,t4
// the trigger list contains t5 and t6
ADD,t5,t6
```

在这个文件中，定义了t1,t2,t3,t4,t5,t6共6个过滤条件，其中t5和t6是通过AND复合的过滤条件。而最后的ADD语句则说明如果新闻满足t5或t6，则该新闻要用户需要关注的新闻。

3.2 过滤的设计

注意到过滤总是针对前面的NewsStory进行的，并且具有多种形式，可以考虑继承的形式实现，例如

```
class Trigger
{
private:
    virtual bool doevaluate(const NewsStory&) const = 0;

public:
    bool evaluate(const NewsStory& story)
    {
        return doevaluate(story);
    }
};

class PhraseTrigger: public Trigger
{
public:
    PhraseTrigger(const std::string& phrase)
    : phrase_{phrase}
    {}

protected:
   bool is_phrase_in(const std::string& text) const;

private:
    bool doevaluate(const NewsStory&) const
    {
        return true;
    }

private:
    std::string phrase_;
};


class TitleTrigger: public PhraseTrigger
{
public:
    TitleTrigger(const std::string phrase)
    : PhraseTrigger(phrase)
    {}
private:
    bool doevaluate(const NewsStory& story) const
    {
        return is_phrase_in(story.Title());
    }
};
```

```
class DescriptionTrigger: public PhraseTrigger
{
public:
    DescriptionTrigger(const std::string phrase)
    : PhraseTrigger(phrase)
    {}
private:
    bool doevaluate(const NewsStory& story) const
    {
        return is_phrase_in(story.Description());
    }
};
```

大家可以设计一下BeforeTrigger和AfterTrigger，以及NotTrigger, AndTrigger和OrTrigger的实现。

3.3 如何在内存中表示所有的过滤条件?

回到3.1中，我们在解析文件时，需要把所有的命名的过滤条件都记录下来，可以使用一个map形式的数据，即

```
std::map<std::string, Trigger*> triggers;
```

而把所有ADD后面的过滤条件放在一个列表TriggerList list中，以便使用。同时，我们定义函数

```
std::istream& read_trigger_config(std::istream& is, TriggerList& list);
```

来解析triggers.txt文件。read_trigger_config的实现有一些难度，可以自己先自行设计一下。

3.4 过滤

过滤的过程非常简单，主体代码如下:

```
    NewsStories stories = fetch();

    TriggerList triggers;
    std::ifstream fs{"triggers.txt"};
    read_trigger_config(fs, triggers);

    for(auto& story: stories){
        for(auto& ptrigger: triggers)
            if (ptrigger->evaluate(story)){
                std::cout << story << "\n";
                break;
            }
    }
```

参考资料

[1] 如何用RSS订阅？ https://zhuanlan.zhihu.com/p/64457116

[2] 中国新闻网。http://www.chinanews.com/rss/

[3] SFML的network功能。https://www.sfml-dev.org/documentation/2.5.1/classsf_1_1Http.php

[4] C++的re功能。https://en.cppreference.com/w/cpp/regex