

第3章 撰写一篇论文

本章提要

在上一章中，我们通过演示一篇论文的前期准备工作，为大家介绍了标题、列表、超链接以及待办事项四种 Markdown 标记的用法。接下来，我们就要开始正式撰写论文了。和之前一样，我们仍会继续以论文写作的过程为导引，逐步深入地介绍其余主要的 Markdown 标记，以及它们的具体使用。这其中既会包含用来表示段落、强调、引用、代码这些基本元素的原生 Markdown 标记，也会涉及到与表格、图形相关的扩展标记，以及它们的基本用法。

3.1 VSCode编辑器

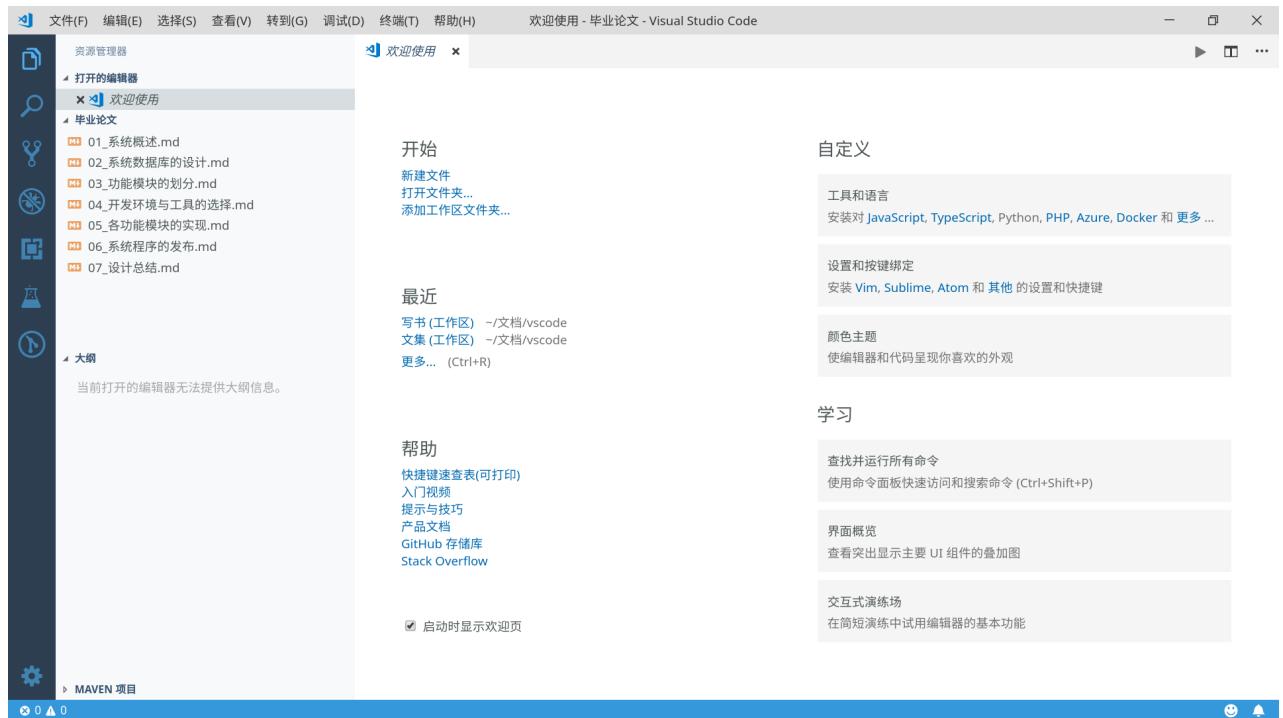
依据我们在上一章中所划分的应用场景，论文的写作应该属于在本地计算机上的编辑工作，而且根据之前拟定的论文大纲来看，我接下来要撰写的是一部可分为七个章节的作品。为了便于后期对作品进行版本控制和审阅修改，我将每一章都存储成了一个独立的 Markdown 文档。鉴于接下来的这部分工作都将通过支持项目管理功能的 VSCode 编辑器来完成。所以，我们要先花一点时间简单介绍一下这款编辑器的基本使用。

在 VSCode 编辑器中，项目管理是以「工作区」为单位来组织的。所以，我们首先要做的第一件事就是创建一个名为「毕业论文」的工作区，其操作步骤如下：

1. 利用本地文件管理系统创建一个名为「毕业论文」的空文件夹，并根据论文大纲中的一级标题创建七个扩展名为 .md 的文本文档，每一个文档对应论文的一个章节：

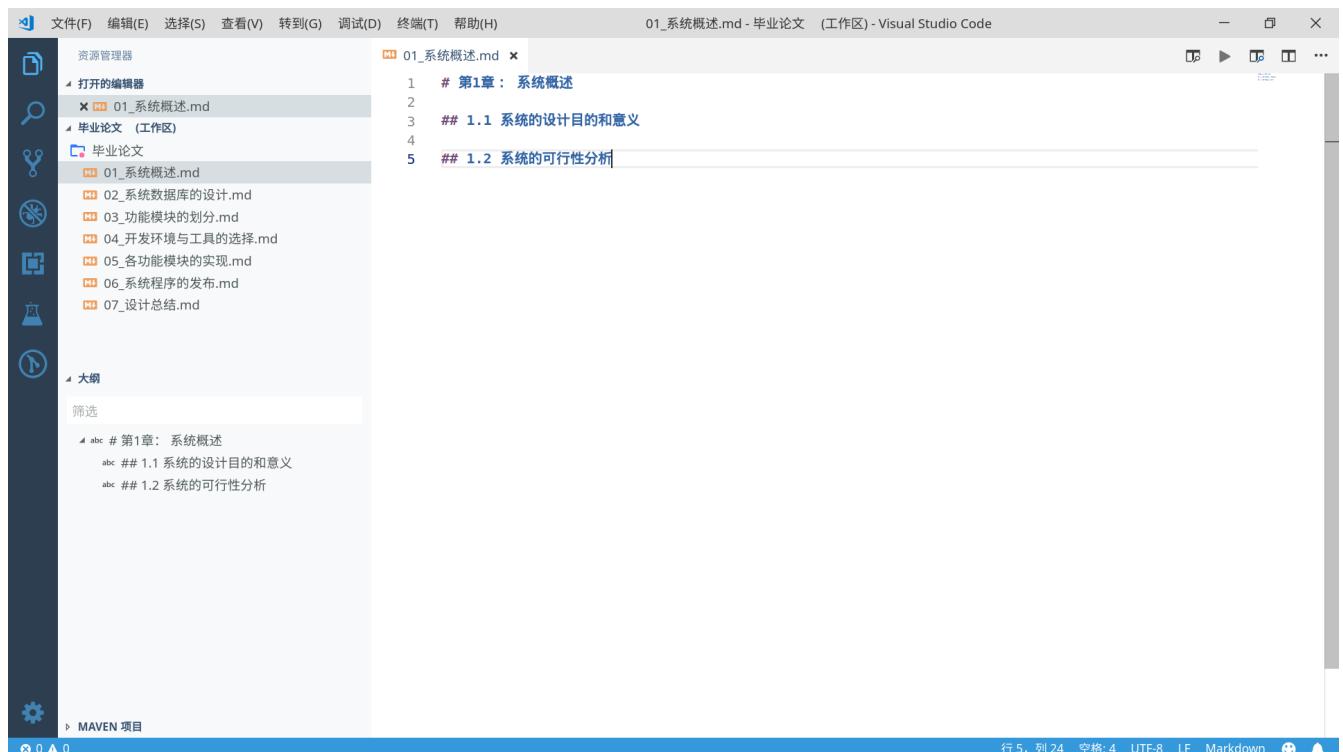


2. 在 VSCode 编辑器中单击菜单「文件 - 打开文件夹...」，打开上面所创建的文件夹：



3. 在VSCode编辑器中单击菜单「文件 - 将工作区另存为...」，保存为「毕业论文」。

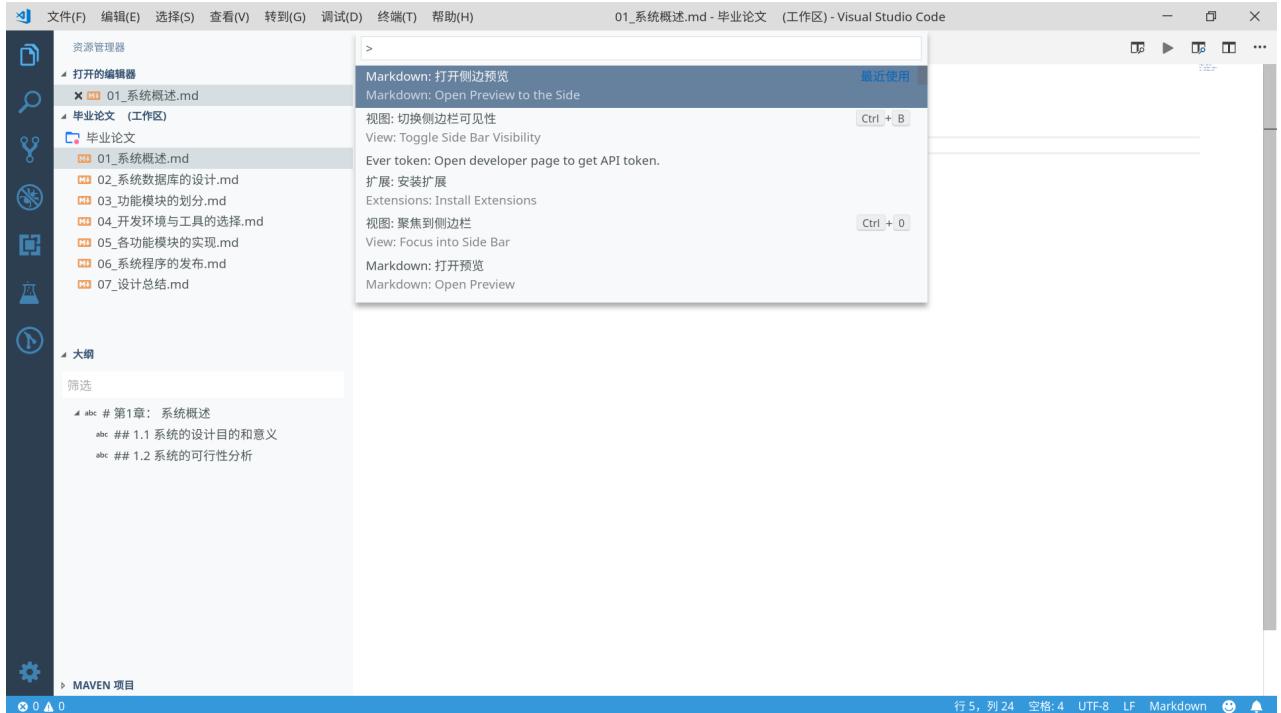
当然了，如果读者使用的是其他本地编辑器（譬如Sublime Text、Atom或Typora等），它们也都有与「工作区」类似的项目管理单位，操作大同小异，读者大可以自行举一反三，做点随机应变即可。接下来，就让我们在VSCode编辑器中打开名为「01_系统概述.md」的文档，将论文大纲中第一章的部分录入，顺便来看一下这款编辑器的布局吧：



如你所见，整个编辑器除去最顶部的菜单栏和最左侧的工具栏，其主体部分主要被分成了三个部分：左侧的上半部分是「文件管理器」，其中显示的是当前工作区中的所有文档；左侧下部分是文档的「大纲视图」，各位可以看到，我们刚刚录入的标题内容已经以大纲的形式同步显示在这一部分了；右侧就是文档的编辑器，这篇论文的 Markdown 编码就是要这一区域中进行。

除此之外，我们在具体编写 Markdown 文档时还可以打开「预览视图」，以实时监控 Markdown 编码的渲染效果。打开「预览视图」的操作非常简单，步骤如下：

1. 在 VSCode 编辑器中使用快捷键 `Ctrl+Shift+P` 打开命令模式，从中选择「Markdown：打开侧边预览」：



然后我们就会在右侧看见当前文档的预览：



2. 如果我们想更专注于当前文档的编写，也可以选择隐藏掉左侧的「文件管理器」和「大纲视图」，只需要用鼠标单击一下最左侧工具栏中的第一个图标，或者使用快捷键 `Ctrl+Shift+P` 打开命令模式，从中选择「视图：切换侧边栏可见性」，然后编辑器的主要区域就只剩下编码区和预览区了：



如果我们想重新打开侧边栏，只需要再次用鼠标单击一下最左侧工具栏中的第一个图标，或者使用快捷键 `Ctrl+Shift+P` 打开命令模式，从中选择「视图：切换侧边栏可见性」即可。

接下来，就让我们正式进入论文的撰写工作吧！

3.2 文字与代码

我们之所以要用 `Markdown` 来写作，而不直接使用无格式的纯文本，主要是为了将作品中的文字标记成各种元素，用这些元素来表达各种不同的语义。因此，我们在使用 `Markdown` 标记时，脑中必须要有一个清晰的概念：我们标记的是语义，不是外观样式。譬如，被标记加粗或倾斜的文字代表的应该是作者对这段文字的着重强调，而不作者认为这段文字加粗或倾斜会比较好看。同样的，被标记为引用的文字它代表的则是作者对其他作品的引述，而不是作者认为某段文字应该有个文本框。

这也就意味着，如果我们想用好 `Markdown` 标记，首先就要先熟悉这些标记所代表的语义。下面，就让我们从最基本的文字元素开始。

3.2.1 段落与换行

在 `Markdown` 中，段落是由一行或多行文本组成的语义元素，对应的是 `HTML` 中 `<p>段落</p>` 标记。该元素的前后必须各有一个以上的空白行，并且段落之间无论有多少个空行，在渲染效果中都只显示一个空行。举个例子，下面这段文字是著名宋词《水调歌头·明月几时有》的原文：

- 1 明月几时有？把酒问青天。不知天上宫阙，今夕是何年？我欲乘风归去，又恐琼楼玉宇，高处不胜寒。起舞弄清影，何似在人间？
- 2 转朱阁，低绮户，照无眠。不应有恨，何事长向别时圆？人有悲欢离合，月有阴晴圆缺，此事古难全。但愿人长久，千里共婵娟。

我们将会看到，虽然上面这段文字是分做两行被录入的（这里不考虑因编辑器自身大小而形成的自动换行），但在渲染效果中，这段文字不仅只显示为一个段落，连我们输入的逻辑换行符也消失了，只有软件界面自身形成的自动换行。

明月几时有？把酒问青天。不知天上官阙，今夕是何年？我欲乘风归去，又恐琼楼玉宇，高处不胜寒。起舞弄清影，何似在人间？转朱阁，低绮户，照无眠。不应有恨，何事长向别时圆？人有悲欢离合，月有阴晴圆缺，此事古难全。但愿人长久，千里共婵娟。

现在，让我们在刚才输入的两行之间再插入一个空行（也可以插入多个空行，效果是一样的）：

```
1 | 明月几时有？把酒问青天。不知天上官阙，今夕是何年？我欲乘风归去，又恐琼楼玉宇，高处不胜寒。起舞弄清影，何似在人间？
2 |
3 | 转朱阁，低绮户，照无眠。不应有恨，何事长向别时圆？人有悲欢离合，月有阴晴圆缺，此事古难全。但愿人长久，千里共婵娟。
```

然后我们就会看到，这一回文字的渲染效果变成了两个段落，对应的是这篇词文的上下阙：

明月几时有？把酒问青天。不知天上官阙，今夕是何年？我欲乘风归去，又恐琼楼玉宇，高处不胜寒。起舞弄清影，何似在人间？

转朱阁，低绮户，照无眠。不应有恨，何事长向别时圆？人有悲欢离合，月有阴晴圆缺，此事古难全。但愿人长久，千里共婵娟。

然而，这种书写方式依然不符合中国古诗词的优雅风格，因为这篇宋词除了分上下阙，通常在阙内也是要分行的，但这种分行不分段的书写形式（即输入 HTML 中的 `
` 标记）在 Markdown 中应该如何实现呢？答案是：在每一行结尾处先输入两个空格，然后再键入回车键：

```
1 | 明月几时有？把酒问青天。          <!--行尾应有两个空格-->
2 | 不知天上官阙，今夕是何年？        <!--行尾应有两个空格-->
3 | 我欲乘风归去，又恐琼楼玉宇，高处不胜寒。 <!--行尾应有两个空格-->
4 | 起舞弄清影，何似在人间？          <!--行尾应有两个空格-->
5 |
6 | 转朱阁，低绮户，照无眠。          <!--行尾应有两个空格-->
7 | 不应有恨，何事长向别时圆？        <!--行尾应有两个空格-->
8 | 人有悲欢离合，月有阴晴圆缺，此事古难全。 <!--行尾应有两个空格-->
9 | 但愿人长久，千里共婵娟。          <!--行尾应有两个空格-->
```

请注意：这里的 `<!-- 注释信息 -->` 是 HTML 中的注释标记，由于兼容性的关系，HTML 标记通常都可以应用在 Markdown 文档中。我们可以用该标记来提示文档在后续处理中需要留意的信息，譬如这里的每个段内分行处都有两个空格，而空格是不可见字符，所以就有必要做个特别提示。接下来，就让我们来看看上述编码的渲染效果吧，这段文字的呈现已经完全符合我们对中国古诗词的一贯书写方式了：

明月几时有？把酒问青天。
不知天上官阙，今夕是何年？
我欲乘风归去，又恐琼楼玉宇，高处不胜寒。
起舞弄清影，何似在人间？

转朱阁，低绮户，照无眠。
不应有恨，何事长向别时圆？
人有悲欢离合，月有阴晴圆缺，此事古难全。
但愿人长久，千里共婵娟。

下面，让我们重新回到论文的撰写工作中，先来看看下面这几段文字的渲染效果：

```
文件(F) 编辑(E) 选择(S) 查看(V) 转到(G) 调试(D) 终端(T) 帮助(H) • 01_系统概述.md - 毕业论文 (工作区) - Visual Studio Code
01_系统概述.md ● 预览 01_系统概述.md x ...
8 以 .NET 技术为代表的 Web 2.0 技术革新的进步意义。
9 ## 1.2 系统的可行性分析
10 因为本系统的设计目的在于学习研究，所以系统设计的可行性主要体现于技术层面的需求和硬件软件方面支持程度两个方面。现分析可行性如下：
11
12 #### 技术可行性
13
14 - ASP：全称 Active Server Pages，是一种以 VBScript 为编程语言的服务器端脚本。特点是易学易用，能快速上手，无需多少学习成本，非常适合本系统这种小型站点的服务器端开发，也能代表比较典型的传统服务器脚本。
15
16 - JavaScript：目前普遍采用的一种浏览器端脚本语言。尽管在一些细节上存在着各种浏览器之间的兼容问题，但市场已经证明了它的可靠性，将其应用于本系统的浏览器端处理也是合适的。
17
18 - HTML：这是所有 Web 技术的基础，可行性无须置疑。
19
20 - Access 数据库：Microsoft Office 组件之一，具有容易获得、功能完整、上手简单、便于移植的特点，适合于本系统的开发和发布。
21
22 #### 硬件的支持
23
24 - 硬件环境：已经购买了一个虚拟主机。本机的设备配置亦能满足程序调试的需求。
25
26 - 软件环境：Windows XP/2003 自带的 IIS 都支持 ASP。且已经安装 Access，数据库连接驱动（ODBC）运作良好。VS2005 的 Web 开发组件调试功能可基于 IIS，并且测试调试接口畅通。软件环境完全可满足需求。
27
28 可行性分析结果：本系统采用的技术和软硬件环境可以满足本系统的设计需求，认为可行。
29
30
31
```

2.0 技术革新的进步意义。

1.2 系统的可行性分析

因为本系统的设计目的在于学习研究，所以系统设计的可行性主要体现于技术层面的需求和硬件软件方面支持程度两个方面。现分析可行性如下：

技术可行性

- ASP：全称 Active Server Pages，是一种以 VBScript 为编程语言的服务器端脚本。特点是易学易用，能快速上手，无需多少学习成本，非常适合本系统这种小型站点的服务器端开发，也能代表比较典型的传统服务器脚本。
- JavaScript：目前普遍采用的一种浏览器端脚本语言。尽管在一些细节上存在着各种浏览器之间的兼容问题，但市场已经证明了它的可靠性，将其应用于本系统的浏览器端处理也是合适的。
- HTML：这是所有 Web 技术的基础，可行性无须置疑。
- Access 数据库：Microsoft Office 组件之一，具有容易获得、功能完整、上手简单、便于移植的特点，适合于本系统的开发和发布。

硬软件的支持

- 硬件环境：已经购买了一个虚拟主机。本机的设备配置亦能满足程序调试的需求。
- 软件环境：Windows XP/2003 自带的 IIS 都支持 ASP。且已经安装 Access，数据库连接驱动（ODBC）运作良好。VS2005 的 Web 开发组件调试功能可基于 IIS，并且测试调试接口畅通。软件环境完全可满足需求。

可行性分析结果：本系统采用的技术和软硬件环境可以满足本系统的设计需求，认为可行。

行 31, 列 1 空格: 4 UTF-8 LF Markdown ⚙️ 📡

如你所见，上图不仅有一般性的文字段落，列表中的每个列表项也可以由一个或多个段落组成。当然，如果某个列表项有多个段落，需要注意一下其第一个段落之后的缩进格式。譬如：

#1 列表中的段落

- 列表项1：这是列表项1的第一个段落，为了更好地呈现段落的状态，我们需要让段落文本的长度达到足以发生自动换行为止，所以会尽可能多地堆砌一些文字，还请见谅！

这是列表项1的第二个段落，为了更好地呈现段落的状态，我们需要让段落文本的长度达到足以发生自动换行为止，所以会尽可能多地堆砌一些文字，还请见谅！
- 列表项2：这是列表项2的第一个段落，为了更好地呈现段落的状态，我们需要让段落文本的长度达到足以发生自动换行为止，所以会尽可能多地堆砌一些文字，还请见谅！

这是列表项2的第二个段落，为了更好地呈现段落的状态，我们需要让段落文本的长度达到足以发生自动换行为止，所以会尽可能多地堆砌一些文字，还请见谅！

但是，如果我们没有做适当的缩进，譬如去掉列表项1的第二个段落之前的空格，大家就会看到该段落脱离了列表，成为了普通的文本段落：

#1 列表中的段落

- 列表项1：这是列表项1的第一个段落，为了更好地呈现段落的状态，我们需要让段落文本的长度达到足以发生自动换行为止，所以会尽可能多地堆砌一些文字，还请见谅！

这是列表项1的第二个段落，为了更好地呈现段落的状态，我们需要让段落文本的长度达到足以发生自动换行为止，所以会尽可能多地堆砌一些文字，还请见谅！

- 列表项2：这是列表项2的第一个段落，为了更好地呈现段落的状态，我们需要让段落文本的长度达到足以发生自动换行为止，所以会尽可能多地堆砌一些文字，还请见谅！

这是列表项2的第二个段落，为了更好地呈现段落的状态，我们需要让段落文本的长度达到足以发生自动换行为止，所以会尽可能多地堆砌一些文字，还请见谅！

列表中的段落

• 列表项1：这是列表项1的第一个段落，为了更好地呈现段落的状态，我们需要让段落文本的长度达到足以发生自动换行为止，所以会尽可能多地堆砌一些文字，还请见谅！

这是列表项1的第二个段落，为了更好地呈现段落的状态，我们需要让段落文本的长度达到足以发生自动换行为止，所以会尽可能多地堆砌一些文字，还请见谅！

• 列表项2：这是列表项2的第一个段落，为了更好地呈现段落的状态，我们需要让段落文本的长度达到足以发生自动换行为止，所以会尽可能多地堆砌一些文字，还请见谅！

这是列表项2的第二个段落，为了更好地呈现段落的状态，我们需要让段落文本的长度达到足以发生自动换行为止，所以会尽可能多地堆砌一些文字，还请见谅！

3.2.2 文字强调

在处理完文本的分段和分行之后，让我们再回头仔细观察一下刚才撰写的论文内容，譬如对于其中的第一个列表：

- ASP：全称 Active Server Pages，是一种以 VBScript 为编程语言的服务器端脚本。特点是易学易用，能快速上手，无需多少学习成本，非常适合本系统这种小型站点的服务器端开发，也能代表比较典型的传统服务器脚本。
- JavaScript：目前普遍采用的一种浏览器端脚本语言。尽管在一些细节上存在着各种浏览器之间的兼容问题，但市场已经证明了它的可靠性，将其应用于本系统的浏览器端处理也是合适的。
- HTML：这是所有 Web 技术的基础，可行性无须置疑。
- Access 数据库：Microsoft Office 组件之一，具有容易获得、功能完整、上手简单、便于移植的特点，适合于本系统的开发和发布。

相信大家都会觉得这段文字看上去难免有些苍白，缺乏重点。而从作者的角度来说，我们肯定会希望能凸显每个列表项的主题以及其中的一些关键字，以便提示阅读时应注意的重点。在这种情况下，我们就需要用到能表达强调语义的元素了。在 Markdown 中，用于表示强调的语义元素主要有三种，分别是粗体显示（对应 HTML 的 **粗体显示** 标记）、斜体显示（对应 HTML 中 *斜体显示* 标记）与粗体+斜体显示。这些语义元素又各自有一到两种不同风格的语法，下面我们就逐一来演示一下这些语法及其渲染效果：

Markdown语法	相应渲染效果
普通文本示例	普通文本示例
粗体文本示例1	粗体文本示例1
<u>粗体文本示例2</u>	粗体文本示例2
<i>斜体文本示例1</i>	<i>斜体文本示例1</i>
<u>斜体文本示例2</u>	<i>斜体文本示例2</i>
<i>粗体+斜体文本示例1</i>	<i>粗体+斜体文本示例1</i>
<u>粗体+斜体文本示例2</u>	<i>粗体+斜体文本示例2</i>
~删除文本示例~	删除文本示例

在使用以上语义元素时，我们应该要注意保持语法的一致性。也就是说，在选定了一种粗体或斜体语法之后，至少要在同一文档或同一作品中坚持使用相同的语法。这种语法上的一致性不仅有利于文档的可读性，也有利于文档的后续批量化处理。现在，就让我们在之前提到的论文文本加入一些用于表示强调的标记吧：

- ```

1 - **ASP**: 全称 Active Server Pages, 是一种以 VBScript 为编程语言的*服务器端脚本*。特点是易学易用，能快速上手，无需多少学习成本，非常适合本系统这种小型站点的服务器端开发，也能代表比较典型的传统服务器脚本。
2
3 - **JavaScript**: 目前普遍采用的一种*浏览器端脚本语言*。尽管在一些细节上存在着各种浏览器之间的兼容问题，但市场已经证明了它的可靠性，将其应用于本系统的浏览器端处理也是合适的。
4
5 - **HTML**: 这是所有 Web 技术的基础，可行性无须置疑。
6
7 - **Access**: Microsoft Office 中的*数据库*组件，具有容易获得、功能完整、上手简单、便于移植的特点，适合于本系统的开发和发布。

```

如你所见，这一回我们用粗体凸显了每个列表项的主题，并用斜体凸显了其中的一些关键字。这样一来，当我们再次查看这段文字的渲染效果时，这段文字给人的感觉就好了不少：

- **ASP**: 全称 Active Server Pages, 是一种以 VBScript 为编程语言的服务器端脚本。特点是易学易用，能快速上手，无需多少学习成本，非常适合本系统这种小型站点的服务器端开发，也能代表比较典型的传统服务器脚本。
- **JavaScript**: 目前普遍采用的一种浏览器端脚本语言。尽管在一些细节上存在着各种浏览器之间的兼容问题，但市场已经证明了它的可靠性，将其应用于本系统的浏览器端处理也是合适的。
- **HTML**: 这是所有 Web 技术的基础，可行性无须置疑。
- **Access**: Microsoft Office 中的数据库组件，具有容易获得、功能完整、上手简单、便于移植的特点，适合于本系统的开发和发布。

### 3.2.3 引用外部文本

在写作过程中，我们难免要引用一些名人名言或其他作品中的某段话，这时候就需要用到能标记出引用文本的元素了。在 Markdown 中，标记应用文本的语法很简单，下面我们就 在有道云笔记 中演示一下：

1 下面是维基百科关于浏览器-服务器结构的定义：

2

3 > 浏览器-服务器 (Browser/Server) 结构，简称B/S结构，与C/S结构不同，其客户端不需要安装专门的软件，只需要浏览器即可，浏览器通过Web服务器与数据库进行交互，可以方便的在不同平台下工作；服务器端可采用高性能计算机，并安装Oracle、Sybase、Informix等大型数据库。B/S结构简化了客户端的工作，它是随着Internet技术兴起而产生的，对C/S技术的改进，但该结构下服务器端的工作较重，对服务器的性能要求更高。

4

5 而对于 Web 浏览器，维基百科中又是这样定义的：

6

7 > 网页浏览器（英语：web browser）：常被简称为浏览器，是一种用于检索并展示万维网信息资源的应用程序。这些信息资源可为网页、图片、影音或其他内容，它们由统一资源标志符标志。信息资源中的超链接可使用户方便地浏览相关信息。

8 >

9 > 网页浏览器虽然主要用于使用万维网，但也可用于获取专用网络中网页服务器之信息或文件系统内之文件。

下面是维基百科关于浏览器-服务器结构的定义：

浏览器-服务器 (Browser/Server) 结构，简称B/S结构，与C/S结构不同，其客户端不需要安装专门的软件，只需要浏览器即可，浏览器通过Web服务器与数据库进行交互，可以方便的在不同平台下工作；服务器端可采用高性能计算机，并安装Oracle、Sybase、Informix等大型数据库。B/S结构简化了客户端的工作，它是随着Internet技术兴起而产生的，对C/S技术的改进，但该结构下服务器端的工作较重，对服务器的性能要求更高。

而对于 Web 浏览器，维基百科中又是这样定义的：

网页浏览器（英语：web browser）：常被简称为浏览器，是一种用于检索并展示万维网信息资源的应用程序。这些信息资源可为网页、图片、影音或其他内容，它们由统一资源标志符标志。信息资源中的超链接可使用户方便地浏览相关信息。

网页浏览器虽然主要用于使用万维网，但也可用于获取专用网络中网页服务器之信息或文件系统内之文件。

如你所见，想要标记一段引用文本，只需要在这段文本的每一行之前加上一个“>”符号即可。请注意：这里所说的“行”，指的是我们输入的逻辑行，包括用于分段的空白行，但不包含软件界面自身形成的自动换行。

除此之外，我们还可以做一些嵌套引用，即我们引用的文本自身也引用了其他地方的文本，譬如：

1 > 自五四运动以来，太多话被塞到了鲁迅的嘴里，于是就有了一个共识：  
2 >> 大家都认为鲁迅说过：  
3 >>> 这句话不是我说的！

我们可以来看看上述编码的渲染效果：

自五四运动以来，太多话被塞到了鲁迅的嘴里，于是就有了一个共识：

大家都认为鲁迅说过：

这句话不是我说的！

当然，引用文本中自然也能使用其他 Markdown 元素，包括标题、列表等。这些都不在话下，使用方式也完全相同，这里就不多累述了。

### 3.2.4 展示程序代码

由于我们正在撰写的是一篇计算机专业的论文，展示程序代码显然是绕不过去的一个环节。在显示代码时，我们最低限度的要求是被标记为代码的文本必须要将其所有字符原样呈现，也就是说，在表示代码语义的元素内，所有 Markdown 的语法标记（譬如“#”、“\*”等字符）都会失效，回归到其字符本身。在 Markdown 中，用来标记代码的语义元素主要有两种：

- **行内代码：**顾名思义，就是将代码显示在普通的文本行之内。这种标记代码的语法就是在代码文本的两侧各加一个“`”符号（输入该符号的按键在普通键盘左侧的第一列键中，位于数字1键的左边）。例如：

1 在C语言中。我们可以通过 `printf("Hello World! ")` 来在终端输出“Hello World!”字样。

其渲染效果如下：

在C语言中。我们可以通过 `printf("Hello World! ")` 来在终端输出“Hello World!”字样。

- **代码区块**: 顾名思义，就是将代码显示在一个单独的区块中，通常用于显示多行代码。标记代码区块的语法就是在代码文本的之前和之后都插入一个只包含三个“`”符号的行，或者对代码文本整体做一个制表符（即四个空格）的缩进。例如：

The screenshot shows a browser window with two code blocks side-by-side.

**Left Side (First Way):**

```

```c
#include <stdio.h>
#include "stack.h"

int main(void)
{
    printf("%s\n", "hello world!");
    return 0;
}
```

```

**Right Side (Second Way):**

```

```c
#include <stdio.h>
#include "stack.h"

int main(void)
{
    printf("%s\n", "hello world!");
    return 0;
}
```

```

另外，为了让 Markdown 阅读器或转换器更好地渲染目标代码的高亮效果，我们通常还会在第一行的三个“`”符号之后声明一下这段代码所使用的编程语言，譬如：

The screenshot shows a browser window with annotated code blocks.

**Left Side (Annotated C Code):**

```

```c
#include <stdio.h>
#include "stack.h"

int main(void)
{
    printf("%s\n", "hello world!");
    return 0;
}
```

```

**Right Side (Annotated Python Code):**

```

```python
#!/usr/bin/env python
# -*- coding: utf-8 -*-

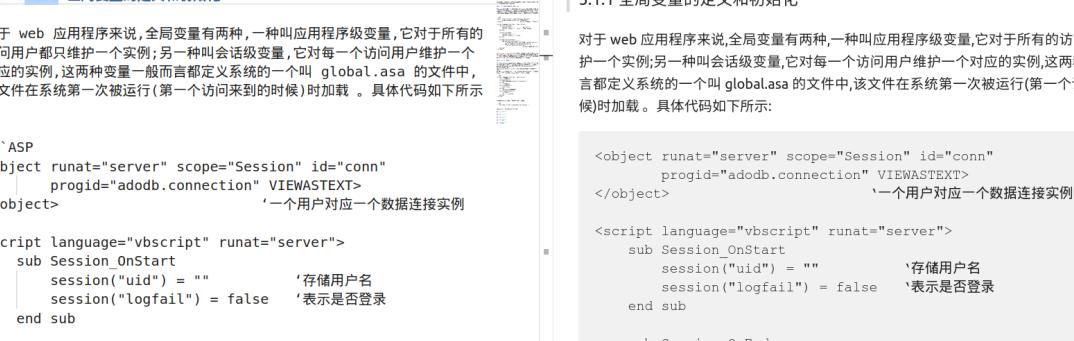
print("hello, world!")
```

```

下面，让我们再次回到论文的撰写工作中来，论述一下系统的实现部分吧，这一回要展示的是古老的 ASP 代码了：

对于 web 应用程序来说,全局变量有两种,一种叫应用程序级变量,它对于所有的访问用户都只维护一个实例;另一种叫会话级变量,它对每一个访问用户维护一个对应的实例,这两种变量一般而言都定义系统的一个叫 global.asax 的文件中,该文件在系统第一次被运行(第一个访问来到的时候)时加载。具体代码如下所示:

```
10 ### 5.1.1 全局变量的定义和初始化
11
12 对于 web 应用程序来说,全局变量有两种,一种叫应用程序级变量,它对于所有的
13 访问用户都只维护一个实例;另一种叫会话级变量,它对每一个访问用户维护一个
14 对应的实例,这两种变量一般而言都定义系统的一个叫 global.asax 的文件中,
15 该文件在系统第一次被运行(第一个访问来到的时候)时加载。具体代码如下所示
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40 其中, `conn`、`session` 集合中的所有变量均为会话级变量,由访问用户各自
```



### 3.2.5 其他文本元素

在写作过程中，我们有时候还会用到一些不太常见，且难以归类的文本元素。在这一节中，我们就来简单介绍一下这些元素。

首先，我们要介绍一些**特殊字符**，这些字符由于 Markdown 自身的语法及其兼容 HTML 的关系，在文本中的某些地方会具有一些特殊作用。在这种情况下，如果我们不想这些字符显示为行内代码，就需要在这些字符前面加上一个反斜杠“\”来显示它们了。这些字符的数量并不多，我们可以在这里将它们罗列一下：

| Markdown语法 | 渲染效果 | 符号说明 |
|------------|------|------|
| \ \        | \    | 反斜杠  |
| \ *        | *    | 星号   |
| \ _        | -    | 下划线  |
| \{ \}      | {}   | 花括号  |
| \[ \]      | []   | 方括号  |
| \( \)      | ( )  | 圆括号  |
| \#         | #    | 井号   |
| \+         | +    | 加号   |
| \-         | -    | 减号   |
| \.         | .    | 英文句点 |
| \!         | !    | 英文叹号 |

除了以上这些特殊字符之外，我们有时候还会用到一个叫做分隔线的文本元素。分隔线有三种不同的符号（分别是星号`*`、减号`-`和下划线`_`）以及下面三种不同的语法风格：

```
1 第一种风格:
2
3 ***
4 ---
5 ___
6
7 当然，如果你想让编码看起来美观一点，也可以在以上任何一种
8 风格的三个符号之间插入一到多个空格，于是就有了第二种风格：
9 * * *
10 - - -
11 - - -
12
13 也可以多使用资格相同的字符，下面是第三种风格：
14
15 *****
16 -----
17 _____
18
```

第一种风格：

当然，如果你想让编码看起来美观一点，也可以在以上任何一种风  
格的三个符号之间插入一到多个空格，于是就有了第二种风格：

也可以多使用资格相同的字符，下面是第三种风格：

当然，这里同样要提醒大家注意保持语法的一致性，当我们选择了一种分割线符号和语法风格之后，请至少在同一文档或同一作品中坚持使用相同的符号和语法风格。这种一致性不仅有利于文档的可读性，也有利于文档的后续批量化处理。

最后，还有一个语义元素也需要稍微关注一下，那就是脚注元素。在撰写文章时，我们总免不了要对文中的某些句子和单词加一些注解。在 Markdown 中，我们可以这样在文章中添加脚注：

```
1 在这个示例段落中，我们打算为大家演示一下如何在 Markdown 文档中添加脚注，以便大家对脚注[^1]这个语义元素
2 [^2]有一个直观的认知。
3
4 * * *
5 [^1]: 这是第一个脚注。
6
7 [^2]: 这是第二个脚注。
```

如你所见，如果我们想对某一文本添加一个脚注，就在该文本后面添加一个与之对应脚注编号（譬如`[^1]`、`[^2]`），然后，在别处另起一个段落，通常会选择在某一部分或整篇文章的结尾处，以该编号加英文冒号为开头（譬如`[^1]:`）来编写脚注的内容。下面，我们来看看上述编码的渲染效果：



在这个示例段落中，我们打算为大家演示一下如何在 Markdown 文档中添加脚注，以便大家对脚注<sup>[^1]</sup>这个语义元素<sup>[^2]</sup>有一个直观的认知。

\*\*\*

[^1]这是第一个脚注。

[^2]这是第二个脚注。

1. 这是第一个脚注。 ↵  
2. 这是第二个脚注。 ↵

## 3.3 表格与图形

在编写一篇计算机专业论文的过程中，我们除了会用到文本和代码相关的基本元素之外，通常还需要绘制各种图表，这就需要涉及到 Markdown 的一系列扩展语法。在这一节中，我们就来逐一介绍这些特殊的语义元素。

### 3.3.1 绘制表格

在 Markdown 中，绘制表格主要有两种方式。首先，针对结构简单的表格， Markdown 提供了一套自己的语法。下面，我们就先来示范一下这种语法：

|   |                           |
|---|---------------------------|
| 1 | 姓名   性别   年龄   婚姻状况       |
| 2 | ----- -----: :---: :----- |
| 3 | 张三   男   25   未婚          |
| 4 | 李四   女   35   已婚          |
| 5 | 王五   男   45   离异          |

下面来逐条解释一下这段编码中所使用的语法标记：

1. “|”是表格的单元格之间的分隔符，而“-”则是表头行与普通表格行之间的分隔符。

2. 表格中每一列文字的对齐方式将由表头行与普通表格行之间的“:”来决定：

- :--- 代表左对齐
- :--: 代表居中对齐
- ---: 代表右对齐
- --- 代表默认状态，依然是左对齐

3. 表格中空格符和“|”的数量只是为了代码看起来整洁美观，对其渲染效果没有什么影响。也就是说，上述表格编码在渲染效果上与下面是一致的：

```
1 | 姓名|性别|年龄|婚姻状况|
2 | -|-:|:-:|:-|
3 | 张三|男|25|未婚|
4 | 李四|女|35|已婚|
5 | 王五|男|45|离异|
```

下面，就让我们具体来看一下上述编码的渲染效果吧：

The screenshot shows a browser window with the URL <https://maxiang.io>. On the left, there is a code editor with the following content:

```
表格语法示例：

姓名	性别	年龄	婚姻状况
张三	男	25	未婚
李四	女	35	已婚
王五	男	45	离异

为了代码看起来整洁美观，我们通常会在表格中填充一定数
量的空格符和“-”：

姓名	性别	年龄	婚姻状况
张三	男	25	未婚
李四	女	35	已婚
王五	男	45	离异
```

On the right, there is a preview area titled "表格语法示例：" which displays the rendered table:

| 姓名 | 性别 | 年龄 | 婚姻状况 |
|----|----|----|------|
| 张三 | 男  | 25 | 未婚   |
| 李四 | 女  | 35 | 已婚   |
| 王五 | 男  | 45 | 离异   |

Below the preview, a note says: "为了代码看起来整洁美观，我们通常会在表格中填充一定数量的空格符和“-”：" followed by another rendered table:

| 姓名 | 性别 | 年龄 | 婚姻状况 |
|----|----|----|------|
| 张三 | 男  | 25 | 未婚   |
| 李四 | 女  | 35 | 已婚   |
| 王五 | 男  | 45 | 离异   |

但是，这种表格标记只能应对简单的表格，对于那些包含跨行或跨列单元格的复杂表格，它就无能为力了。这时候，我们就需要利用 `Markdown` 对 `HTML` 的兼容性，使用 `HTML` 的 `<table>` 标签来完成这种复杂的表格了，譬如：

```
1 <table>
2 <tr>
3 <th rowspan=4>会员登记表</th>
4 <th>姓名</th>
5 <th colspan=3>个人信息</th>
6 </tr>
7 <tr>
8 <td>张三</td><td>男</td><td>25</td><td>未婚</td>
9 </tr>
10 <tr>
11 <td>李四</td><td>女</td><td>35</td><td>已婚</td>
12 </tr>
13 <tr>
14 <td>王五</td><td>男</td><td>45</td><td>离异</td>
15 </tr>
16 </table>
```

如你所见，我们在这里利用 `<td>` 标签的 `colspan` 和 `rowspan` 属性实现了表格中单元格的跨行和跨列，其渲染效果如下：

\*\*绘制复杂表格:\*\*

```
<table>
<tr>
<th rowspan=4>会员登记表</th>
<th>姓名</th>
<th colspan=3>个人信息</th>
</tr>
<tr>
<td>张三</td><td>男</td><td>25</td><td>未婚</td>
</tr>
<tr>
<td>李四</td><td>女</td><td>35</td><td>已婚</td>
</tr>
<tr>
<td>王五</td><td>男</td><td>45</td><td>离异</td>
</tr>
</table>
```

当然，这也就意味着，我们在设计网页时用于绘制表格的那些技巧也都可以运用在这里，但请注意，使用 Markdown 写作的目的之一，就是希望将表现语义的元素与表现样式的元素分而治之。所以，我们在 Markdown 中使用 HTML 标签时也应该坚持这一原则，尽量只专注于内容的表达，而对于内容的呈现样式则尽量交由后续工作来处理（譬如上图中的“会员登记表”这五个字，似乎应该设定为在单元格中纵向显示）。

在掌握了这两种绘制表格的方式之后，我们就可以回到论文中去整理那些表格了，譬如：

```
19 ## 2.2 数据库中各表的设计
20
21 `Users`表用来存储用户注册的基本信息，具体结构如表2-2所示：
22
23 ### 表2-2. 用户基本信息表(users)
24
25 | 字段名 | 数据类型 | 说明 | 备注 |
26 |-----|-----|-----|-----|
27 | Userid | 数字(自动编号) | 用户ID | 主键 |
28 | Username | 文本 | 用户名 | 非空、唯一索引 |
29 | Usrpwd | 文本 | 用户密码 | 非空 |
30 | Usrkey | 文本 | 找回密码的口令 | 非空 |
31 | Usrrealname | 文本 | 真实姓名 | 非空 |
32 | Usrsex | 文本 | 性别 | 仅限男、女二值 |
33 | Usrtel | 文本 | 电话 | |
34 | Usremail | 文本 | E-mail | |
35 | Usraddress | 文本 | 地址 | |
36 | Usrzip | 文本 | 邮编 | |
37 | Usrstate | 文本 | 省份 | |
38 | Usrcity | 文本 | 城市 | |
39
40 此表用来保存用户注册时填写的基本而必要的信息。其中，`Userid`字段是由系统成功写入记录时系统自动编号，作为`users`表的主键为其他关联表提供连接的主要索引。除此之外，其他表也可以通过`Username`字段关联`users`表。此字段的更主要功能是用户登录，以及登录后作为一个用户的唯一标识符存储在该用户建立的会话变量中(即`cookie`的内容)，这点在后面的系统详细设计中会进一步介绍。
41
42 `books`表用来存储一本书的基本而必要的信息，具体结构如表2-3所示：
43
44
45 ### 表2-3. 书籍基本信息表(books)
46
47 | 字段名 | 数据类型 | 说明 | 备注 |
48 |-----|-----|-----|-----|
```

2.2 数据库中各表的设计

`Users`表用来存储用户注册的基本信息，具体结构如表2-2所示:

表2-2. 用户基本信息表(users)

字段名	数据类型	说明	备注
Userid	数字(自动编号)	用户ID	主键
Username	文本	用户名	非空、唯一索引
Usrpwd	文本	用户密码	非空
Usrkey	文本	找回密码的口令	非空
Usrrealname	文本	真实姓名	非空
Usrsex	文本	性别	仅限男、女二值
Usrtel	文本	电话	
Usremail	文本	E-mail	
Usraddress	文本	地址	
Usrzip	文本	邮编	
Usrstate	文本	省份	
Usrcity	文本	城市	

此表用来保存用户注册时填写的基本而必要的信息。其中，`Userid`字段是由系统成功写入记录时系统自动编号，作为`users`表的主键为其他关联表提供连接的主要索引。除此之外，其他表也可以通过`Username`字段关联`users`表。此字段的更主要功能是用户登录，以及登录后作为一个用户的唯一标识符存储在该用户建立的会话变量中(即`cookie`的内容)，这点在后面的系统详细设计中会进一步介绍。

### 3.3.2 引用外部图形

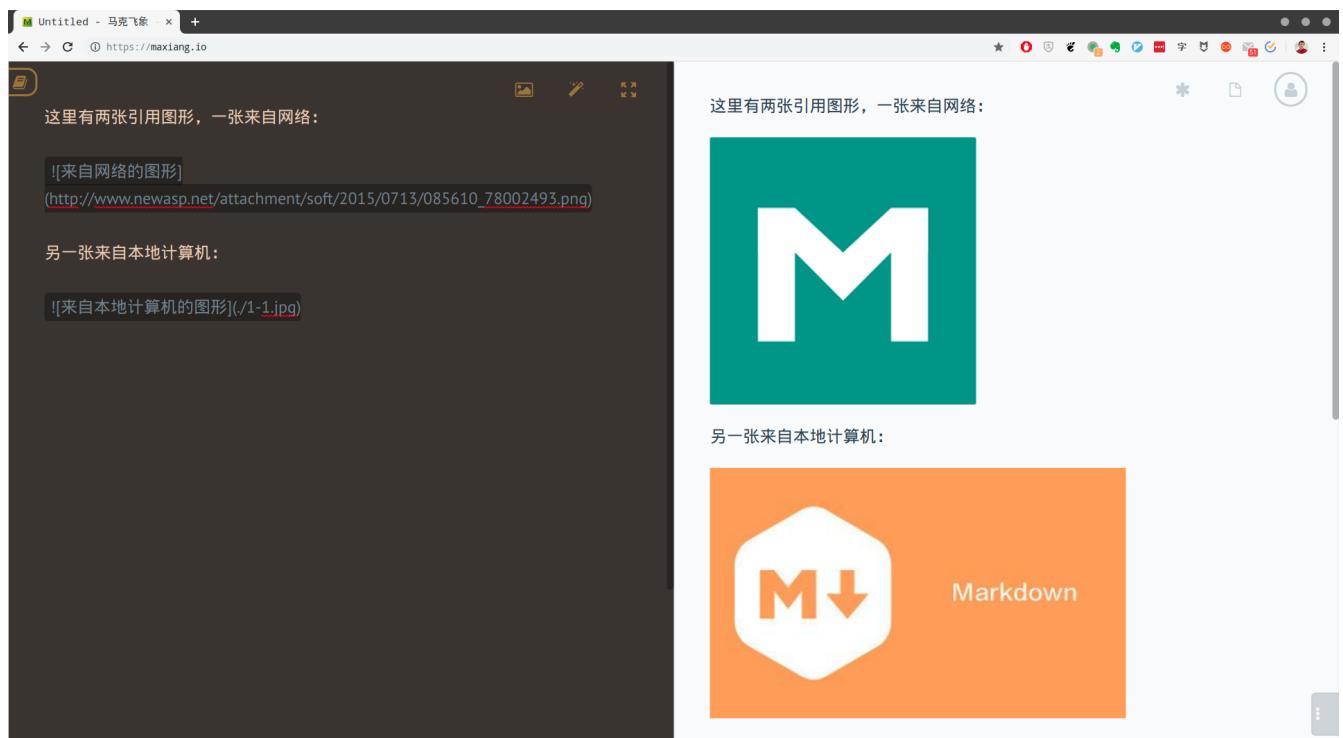
在写作计算机专业论文的过程中，我们通常会需要用到各种各样的图形，以弥补文字在论述能力上的不足。按照图形的来源，我们大致上可以将写作过程中会用到的图形两大类：首先是引用图形，这类图形来自论文外部，包括实物照片、屏幕截图以及第三方制作的图表等，这也是我们最常用的一种在文章中插入图形的方式。在 Markdown 中，引用外部图形的语义元素对应的是 HTML 的 `<img>` 语法，它的语法和网页超链接非常类似，即：

```
1 | ! [引用图形的文字说明] (引用图形的URL)
```

在这里。`引用图形的URL`既可以是一个HTTP协议的网络URL，也可以是一个绝对路径或相对路径形式的本地URL。譬如：

```
1 | 这里有两张引用图形，一张来自网络:
2 |
3 | ! [来自网络的图形] (http://www.newasp.net/attachment/soft/2015/0713/085610_78002493.png)
4 |
5 | 另一张来自本地计算机:
6 |
7 | ! [来自本地计算机的图形] (./1-1.jpg)
```

其渲染效果如下：



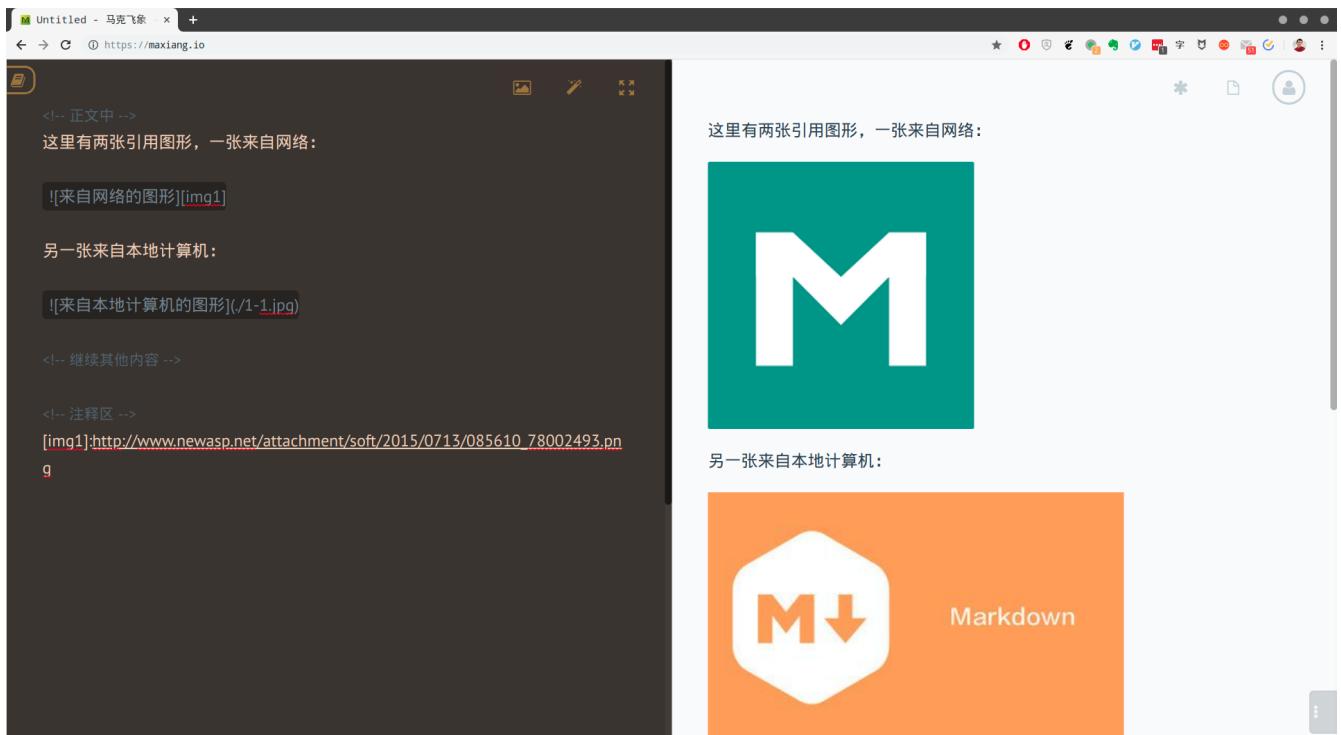
当然，如果我们觉得URL部分的字符串过长，影响文档的整体美观，也可以采用另一种分离式的引用语法。即在图形引用之处为该URL设置一个别名，然后将该别名所代表的URL写在文档最后的注释区即可。譬如，对于上面那张来自网络的图形，我们还可以这样编码：



```

1 <!-- 正文中 -->
2 这里有两张引用图形，一张来自网络：
3
4 ! [来自网络的图形] [img1]
5
6 另一张来自本地计算机：
7
8 ! [来自本地计算机的图形] (./1-1.jpg)
9
10 <!-- 继续其他内容 -->
11
12 <!-- 注释区 -->
13 [img1]:http://www.newasp.net/attachment/soft/2015/0713/085610_78002493.png
```

我们可以看到，上述编码的渲染效果与之前是完全一致的：



接下来，我们再来看看如何在 `Markdown` 中用特定代码自动生成图形。

### 3.3.3 自动生成图形

虽然在大多数情况下，我们都更倾向于在文章中直接引用外部图形，但在一些特定情况下，我们也会需要用到一些少量的生成图形。这里所谓的生成图形，指的是我们在文章中直接使用某种特定代码自动生成的图形，这方面最常见的例子就是各种UML图，譬如流程图、序列图、甘特图等。在 `Markdown` 的诸多扩展中，`PlantUML` 和 `Mermaid` 无疑是目前使用最多的，用于自动生成图形的两款扩展库。在这里，我们打算以 `Mermaid` 为例来介绍一下如何在 `Markdown` 文档中自动生成图形。如果有读者希望使用其它类似扩展库（譬如 `PlantUML`<sup>1</sup> 或 `flowchart.js`<sup>2</sup> 等），也不必担心，这些扩展库在使用方法上都大同小异，只要学会了其中一个，其它都只需要举一反三即可。

`Mermaid` 是一个用于自动生成流程图、序列图、甘特图等UML图形的 `JavaScript` 程序库。正是由于这是一个 `JS` 库，所以它实际生成的并不是真正的“图形”，而是一段 `HTML` 代码，这让它可以更好地融入 `Markdown` 文档中。当然，我们在这里介绍的只是 `Mermaid` 的基本用法，让读者对如何在 `Markdown` 文档中自动生成图形有一个总体上的认知。如果想更深入更全面地学会这个库，还请读者自行查阅 `Mermaid` 库的官方文档<sup>3</sup>。

### 3.3.3.1 生成流程图

众所周知，流程图实际上是对工作流的一种图形化表示，主要用于对程序的控制流和数据流进行建模，是最常用的UML行为图之一。所以，既然是要对工作流进行建模，那么定义流程图的第一步就是要先确定流程的走向。在Mermaid中，描述流程图走向的语法如下：

```
1 | ````Mermaid
2 | graph [流程图的走向]
3 | [其他描述语句...]
4 | ````
```

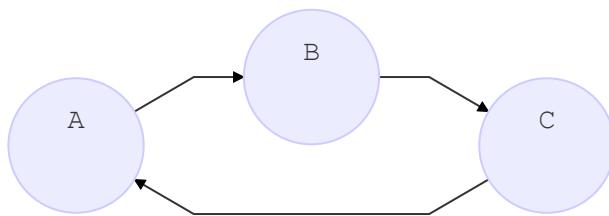
首先，要想在Markdown文档中使用Mermaid库生成图形，我们需要将相关的代码区块标识为“Mermaid”，这样文档的阅读器或转换器才能识别出这是一段Mermaid代码，并调用相关引擎来生成图形。然后，对于“[流程图的走向]”，我们可以有以下四种选择：

走向描述	具体说明
LR	从左到右
RL	从右到左
TB	从上到下
BT	从下到上

下面，我们继续来关注“[其他描述语句...]”的部分。在这一部分的工作中，首先要做的是定义流程图中的“节点”，这些“节点”代表了工作流中的开始、结束、决策、动作等要素。在Mermaid中，每个“节点”都需要设置一个id，以及节点的形状与其中的文字，譬如：

```
1 | ````Mermaid
2 | graph LR
3 | id1((A)) --> id2((B))
4 | id2 --> id3((C))
5 | id3 --> id1
6 | ````
```

其渲染效果如下：



在这里，`id1`、`id2`、`id3`分别是流程图中三个“节点”的id，而关于这些“节点”的形状，我们可以有下面几种选择：

节点类型	相关说明
id[文字]	矩形节点
id(文字)	圆角矩形节点
id((文字))	圆形节点
id>文字]	右向旗帜状节点
id{文字}	菱形节点

请注意：如果“节点”中的文字中包含标点符号，需要用双引号把整段文字括起来。如果希望在文字中使用换行，请使用“`<br/>`”来替代换行。

在定义完“节点”之后，接下来就需要定义这些“节点”之间的“连线”了，在这部分，我们同样可以有以下几种选择：

连线类型	相关说明
>	添加尾部箭头
-	不添加尾部箭头
--	单线
--text--	单线上加文字
--	粗线
--text--	粗线加文字
-.-	虚线
-.text.-	虚线加文字

除此之外，我们还可以在流程图中设置一个子图，子图的语法如下：

```

1 ````Mermaid
2 graph [流程图的走向]
3 [父图描述语句...]
4 subgraph [子图名称]
5 [子图描述语句...]
6 end
7 ````
```

现在，让我们综合一下刚才学到的语法，试着根据论文中要构建的系统来画一张流程图吧，例如在编辑器中输入如下代码：

```

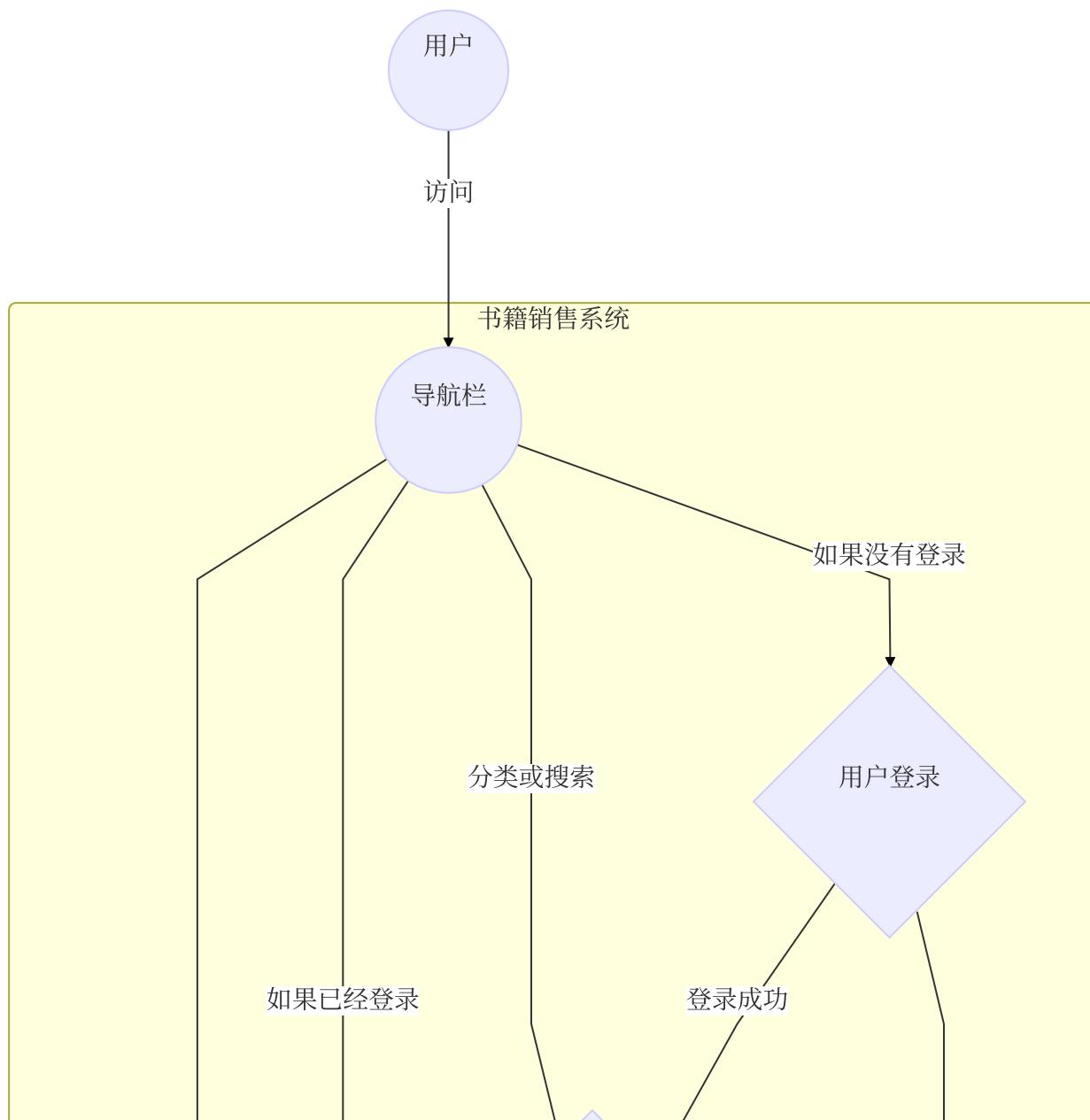
1 ````Mermaid
2 graph TD
3 id1((用户)) --访问--> id2((导航栏))
4 subgraph 书籍销售系统
```

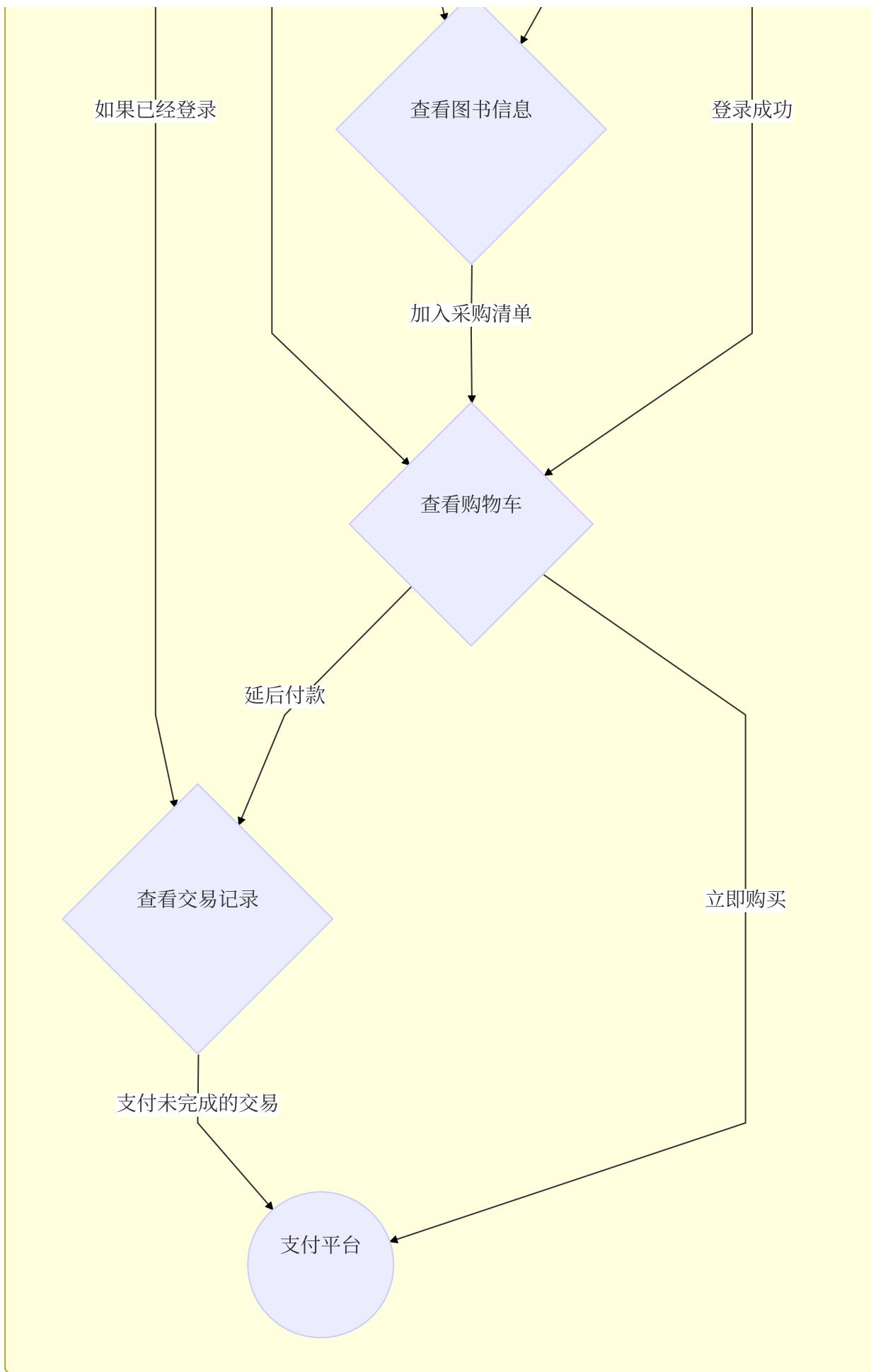
```

5 id2 --分类或搜索--> id3{查看图书信息}
6 id2 --如果没有登录--> id4{用户登录}
7 id2 --如果已经登录--> id5{查看交易记录}
8 id2 --如果已经登录--> id6{查看购物车}
9 id3 --加入采购清单--> id6
10 id4 --登录成功--> id3
11 id4 --登录成功--> id6
12 id5 --支付未完成的交易-->id7((支付平台))
13 id6 --立即购买--> id7
14 id6 --延后付款--> id5
15 end
16 ```

```

只要我们安装了与 `Mermaid` 相关的预览插件（譬如VSCode编辑器的Mermaid Preview插件），应该就会看到如下效果：





### 3.3.3.2 生成序列图

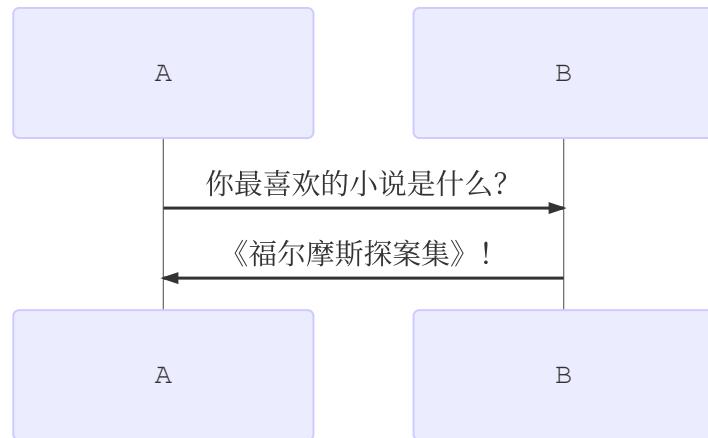
序列图也属于UML行为图中的一种，它的主要作用是描述既定活动序列上的交互行为，通常用来表示相关某一活动中 的动作顺序，它的每一条消息都对应了一个动作或者触发状态转换的事件。在 Mermaid 中，定义序列图的语法如 下：

```
1 ````Mermaid
2 sequenceDiagram
3 [参与者1] [消息线] [参与者2]:[消息文本]
4 ...
5 ````
```

例如，假设我们想描述一个对话序列，就可以这样：

```
1 ````Mermaid
2 sequenceDiagram
3 A ->> B: 你最喜欢的小说是什么?
4 B ->> A: 《福尔摩斯探案集》!
5 ````
```

其渲染效果如下：



在这里，A和B都是该对话序列的“[参与者]”。当然，这是一个最简单的序列图，如果某活动序列中有多个“[参 与者]”，我们也可以单独用 participant 关键字来标明它们，譬如：

```

1 ````Mermaid
2 sequenceDiagram
3 participant A
4 participant B
5 participant C
6 ...
7 ````
```

而且，我们还可以采用 `participant [别名] as [参与者]` 的形式，为相关参与者设置别名，譬如：

```

1 ````Mermaid
2 sequenceDiagram
3 participant A as Alice
4 participant B as Bill
5 participant C as Chioe
6 ...
7 ````
```

同样地，在定义完“`[参与者]`”之后，我们还需要定义它们之间的“`[消息线]`”。对于“`[消息线]`”，`Mermaid`为我们提供了以下几种选择：

消息线类型	具体说明
->	无箭头的实线
-->	无箭头的虚线
->>	有箭头的实线
-->>	有箭头的虚线
-x	末端为叉的实线（表示异步）
--x	末端为叉的虚线（表示异步）

最后一部分是每一个序列的“`[消息文本]`”，这代表的是序列参与者之间所传递的消息内容，譬如在上面的例子中，A 抛给B的消息是“`你最喜欢的小说是什么？`”，而B回复给A的消息则是“`《福尔摩斯探案集》！`”。

除了以上这些序列图的基本元素之外，我们还可以在序列图中加入一些更详细的信息。譬如标记相关消息的处理状态：

- 在消息线末尾增加 `+`，则消息接收者进入当前消息的“处理中”状态。
- 在消息线末尾增加 `-`，则消息接收者离开当前消息的“处理中”状态。

或者直接用以下语法来标记明某个参与者进入“处理中”状态：

```

1 ````Mermaid
2 activate [参与者]
3 ````
```

再譬如，我们可以在某些指定位置上添加一些标注，其语法如下：

```
1 | ````Mermaid
2 | Note [标注位置] [参与者]: [标注文字]
3 | ````
```

在这里，对于“[标注位置]”的描述，我们可以有以下几种选择：

位置描述	具体说明
right of	在参与者的右侧
left of	在参与者的左侧
over	在当中，可以横跨多个参与者

最后，我们还可以在序列图中标记一些控制消息传递的控制结构，其语法如下：

- **循环结构：**

```
1 | ````Mermaid
2 | loop [循环条件]
3 | [循环体描述语句]
4 | end
5 | ````
```

- **条件判断：**

```
1 | ````Mermaid
2 | alt [条件1 描述]
3 | [分支1 描述语句]
4 | else [条件2 描述] # else 分支可选
5 | [分支2 描述语句]
6 | else ...
7 | ...
8 | end
9 | ````
```

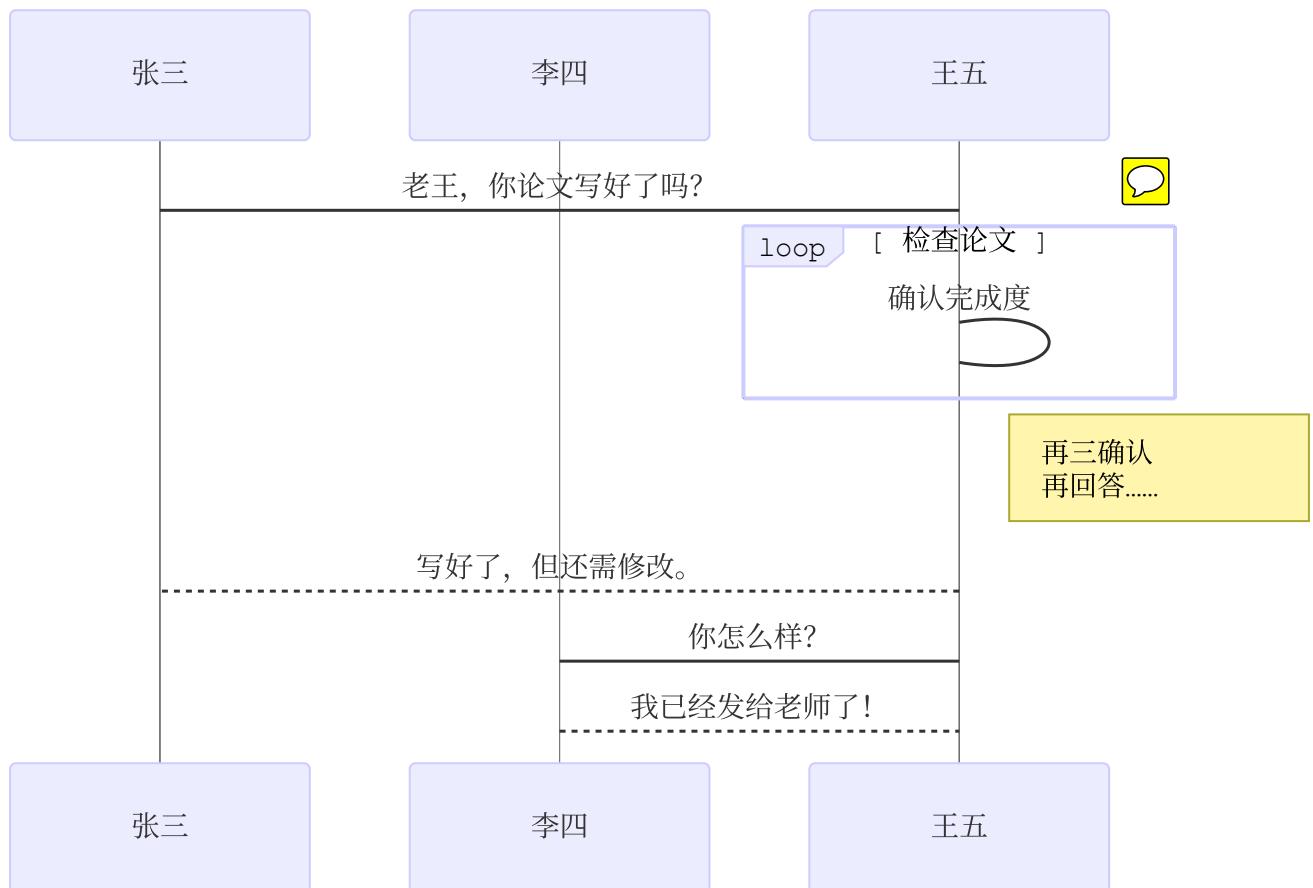
最后，和之前一样，让我们综合一下上面的语法，试着根据写论文的情境来画一张序列图吧。例如在编辑器中输入如下代码：

```
1 | ````Mermaid
2 | sequenceDiagram
3 | participant 张三
4 | participant 李四
5 | participant 王五
6 | 张三 -> 王五: 老王，你论文写好了吗？
7 | loop 检查论文
8 | 王五 -> 王五: 确认完成度
9 | end
10 | Note right of 王五: 再三确认
再回答.....
11 | 王五 --> 张三: 写好了，但还需修改。
12 | 王五 -> 李四: 你怎么样？
```

13 | 李四 --> 王五：我已经发给老师了！

14 | ...

同样地，只要我们安装了与 Mermaid 相关的预览插件，应该就会看到如下效果：



## 本章小结

在这一章中，我们首先介绍了如何用VSCode编辑器创建一个 Markdown 文档项目。然后，我们从组织论文的文本切入，依次介绍了如何在 Markdown 文档中进行文本的分段和换行、强调重点、引用外部文字和图片、展现程序代码以及绘制表格。最后，我们还以用 Mermaid 库生成流程图和序列图为例，介绍了如何在 Markdown 文档中自动生成图形。

到目前为止，我们已经介绍了用 Markdown 撰写一篇计算机专业的大学论文所需的大部分基本技能，唯一尚未提及的是关于数学问题的讨论，这正是我们在下一章中要讨论的问题。

1. 注释：官方网站：<http://plantuml.com/zh/>

2. 注释：官方网站：<http://flowchart.js.org/>

3. 注释：官方网站：<https://mermaidjs.github.io/>