**《软件设计与开发实践A》**

**课程报告**

**任务名称：可视化笔记工具**

**班 级： 组号：计科八班第十七组**

**学 号：190110830**

**姓 名：刘靖宇**

**学年学期：2020年秋季**

**任课教师：刘洋**

哈尔滨工业大学（深圳）计算机科学与技术学院

前 言

《软件设计与开发实践A》是基于自选项目的实践训练，学生将综合利用《集合论与图论》、《数据结构》、《算法设计与分析》、《高级语言程序设计I及II》等方面的基本概念、原理、技术和方法，开展实际应用问题设计求解和对应系统软件开发两大方面的实践。

通过本课程的学习、训练和实践，引导学生熟练掌握问题设计求解和软件编程开发的一般过程、相关技术、方法和途径；训练综合运用所学的理论知识和方法独立分析和解决问题，提高问题分析、问题求解和软件开发能力；培养学生能够针对实际问题，选择适当的数据结构、设计有效算法，提高程序设计的能力和编码质量；训练和学会用系统的观点和软件开发一般规范进行软件设计开发，培养软件工作者所应具备的科学工作方法和作风，提高工程素质；并通过采用团队协作、构建项目组的形式，来培养学生的团队合作与交流能力。

本课程要求学生分组进行（每组1~3人），通过一定的调研来自行结合实际应用需求来选题，并由任课教师来对学生选题做筛选评定。要求所设计开发的软件具有一定的实用性和系统完整性，要有较友好的图形交互操作界面，并对输入数据有较强的完整性约束，要以用户需求作为出发点来设计软件界面和功能模块。本课程主要教学环节包括：学生自选任务、软件开发、软件验收、任务报告撰写提交和任务资料整理归集等。

|  |
| --- |
| 教师评语：                报告成绩： |

1. 选题背景与应用意义

（描述选题的背景、所针对的具体实际问题及任务所体现的实用性价值和意义所在等）

1. 选题背景:

随着信息媒介的不断进步,信息的转播速度达到了空前未有的快速.得益于信息的高速传播,各式各类的信息真真假假,目接不暇,于是从繁杂的信息海洋中提炼出自己的知识库已经成为了提升自我的必经途径.因此,就如何管理自己的知识笔记已经出现了各式的管理方法.

其中最常见的是"文件夹管理法"与标签法".

在介绍各类知识笔记管理方法之前,先介绍知识的结构关系:

{

"单个知识内容不是静止的,而是流动的",对于同一个知识点,他可能出现在

各个不同的体系中.比如基础的集合论知识,他不仅是我们在学习图论是的基本知识,也是我们学习概率论的基本知识,更是通向一切统计分析的概念基础.所以知识的结构,更类似于wiki的网状结构--各类知识相互交融 ,配合形成新的内容(在wiki当中,知识的网状组合是以无数的超链接构成的) .

}

接下来,将介绍各类传统的笔记方法的弊端:

{

文件夹管理的方式弊端非常明显，由于一篇笔记只能存放在一个文件夹下，每个知识点被孤立了起来,所以它严重限制了一篇笔记被发现的几率，导致用户常常遗忘一些低频知识点,并且阻碍了知识的碰撞和创意或者灵感的产生。

跟它明显对比的就是「标签」管理法:

标签管理的方式看似比较合理，一篇笔记可以根据实际的分类打上不同的标签，这样提高了笔记被发现的几率。但是仍然存在一些问题：

通过标签找到一篇笔记后，如果我们顺着这个笔记的其他标签继续往下寻找新的笔记，往往会迷失了方向，很难能够原路返回（当然这一点不绝对）。

标签管理另一个弊端是，当你有 3 年以上的笔记存储，并且打了几千个标签后，它将成为笔记管理的噩梦，没人能清楚都有什么标签以及如何去管理它们。

}

为了解决这些问题,基于 Zettelkasten的笔记管理方法的各类工具应运而生,其核心理念在于把各个知识点分为不可再不可分割的block,并通过双向链接把各个知识块组织起来.(类似维基百科的组织方式),从而使各个内容以更合理的逻辑组织互联,便于管理者的再次消化和二次创造.

}

1. 具体使用场景:

用户进行个人笔记编辑,组织和管理.

1. 实际价值和意义:

总体而言, 为大体量的个人知识库提供了一种更加合理的管理与编辑方案.

1. 以HTML为编辑载体,提供了更为现代,更有兼容性的笔记编辑格式,并支持和markdown文件相互转换.

2. 同时提供了对知识块的可视化的组织方式,可以在一定量的知识块中以不同的逻辑组织,并提供类思维导图的表现方式.

3.支持各个文本块之间以超链接的方式相互链接,并以图的方式展示整个链接网络.

1. 实际问题:

解决传统的笔记方法在解决大体量的知识笔记时的一系列不合理问题(详见背景),为大体量的个人知识库提供了一种更加合理的管理与编辑方案.

1. 需求分析

（根据任务选题的要求，充分地分析和理解问题，明确用户要求做什么？完整性约束条件是什么？运行环境要求、图形操作界面要求等）

1. 运行环境
   1. 基于windows10 的本地系统.

根据我们的开发环境与大多数笔记用户的使用平台,我们打算先在windows 上完成本地功能开发,之后如果还有空余时间,使用qt将其打包成多个跨平台版本,并把用户笔记部署到服务器上,从而实现多平台共享.

1. 功能需求
   1. 现代的文本编辑功能

对文本样式编辑的需求是用户对所有文本编辑器最根本的要求.

介于当今环境下xml格式的文本在各大社交平台的分享不方便,我们与大多数主流笔记软件(evernote , notion, onenote等)一样,采用富文本标记语言,html与md 为笔记样式.在进行样式编辑时,大量采用快捷键,减少用户使用鼠标的次数.

* 1. 多样的格式兼容和导出

介于当下电子设备的多样,在不同使用场景下,对文本的格式有不同的要求(例如md格式的文档在打印时便十分不方便),我们选定了HTML Md Pdf txt ,为主要的格式,其中除pdf不支持编辑之外,其他文件可以互相转换导出,从而便于用户更舒适的在各个使用场景间切换.

* 1. 完备的搜索功能

在整理对大量的毕竟进行管理之时,让用户一个一个的翻找文件夹时极为不合理的,我们的搜索功能包括,1. 在文件管理时进行文件,文件夹的搜索,2. 在编辑文本时对文本内容进行搜索.

* 1. 稳定完善有逻辑的文件管理

虽然windows 自带有较为完善的资源管理器,但用户并不希望在使用笔记软件时还要再笔记软件和windows资源管理器之间跳转.

* 1. 丰富的思维导图编辑功能

我们的核心功能点便是让用户对每一篇笔记进行思维导图对应的编辑,从而帮助用户更好理清文章脉络.

* 1. 较为方便的文本与思维导图的映射

在用户使用思维导图时,往往需要对找笔记文本进行参照,于是我们想构建一个较为简便的方法,让用户可以轻松的在笔记和思维导图之间构建,映射.

1. 界面需求

整体界面尽量符合现代软件的审美,尽量简洁.

系统主要功能设计

（根据需求分析来详细设计软件系统的主要功能模块，要求画出功能模块图，含子功能模块图，并给出详细文字描述）

该项目主要分为三个组件:

搜索和文件管理组件:{

1.基本的文件夹,文件名,块名 搜索

2.单个文件文本内容的匹配搜索.

3.文件夹,文件,的新建 删减

4.文件的导入

}

富文本编辑组件:{基于qt webengine的富文本标记

1.以'/'为功能键扩展的一系列文本标记:{

1.基本的html 文本tag (p h1 h2 h3 li ul )编辑

2.分块功能

3.超链接{

1.块搜索组件

2.预览功能

}

4. 文本内部搜索

}

2.文本的导出(html md '//pdf....')

}

导图映射组件:{

1.导图和文本块的可视化映射

2.导图的删改与保存

3. 导图编辑:{

1.导图的新建

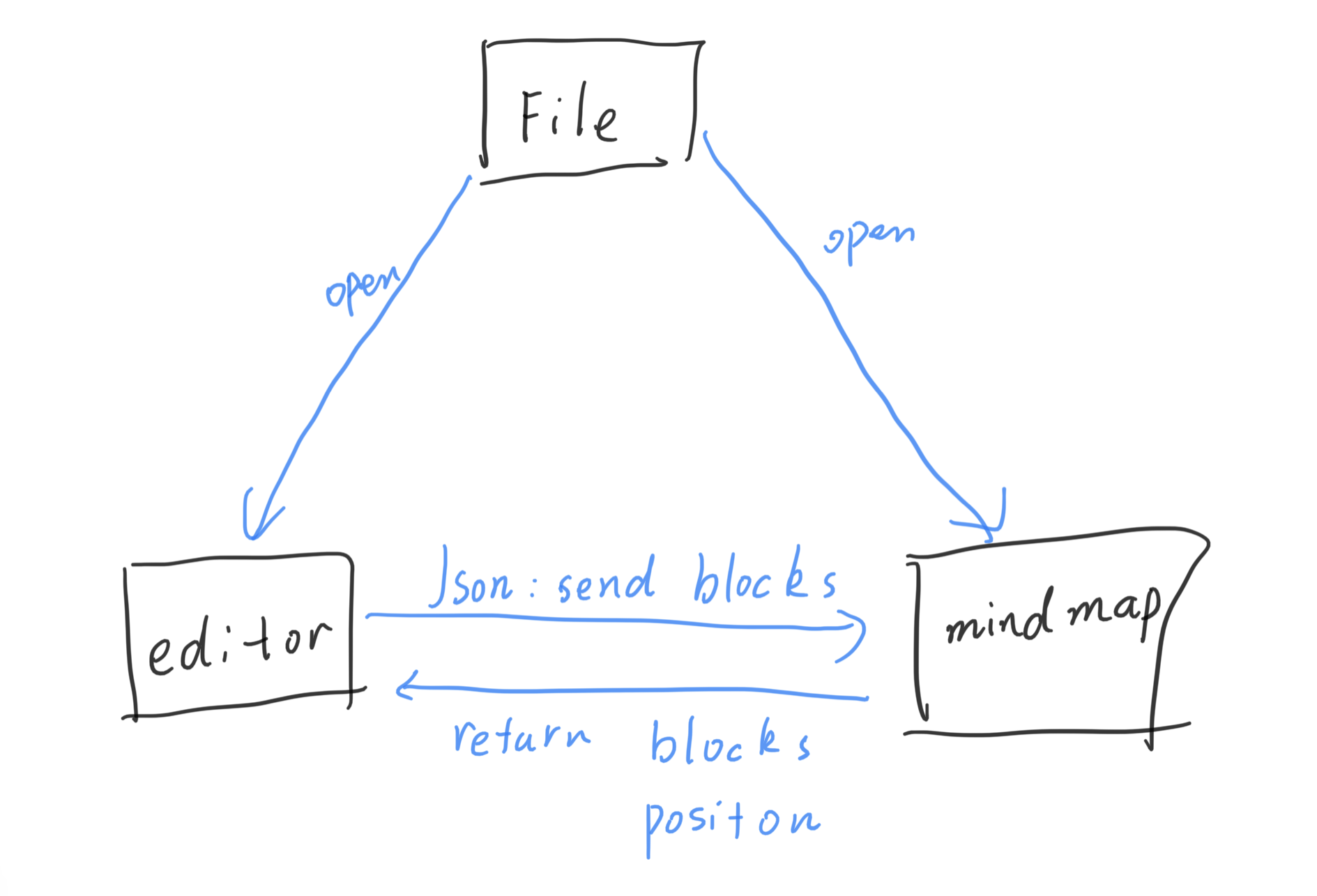
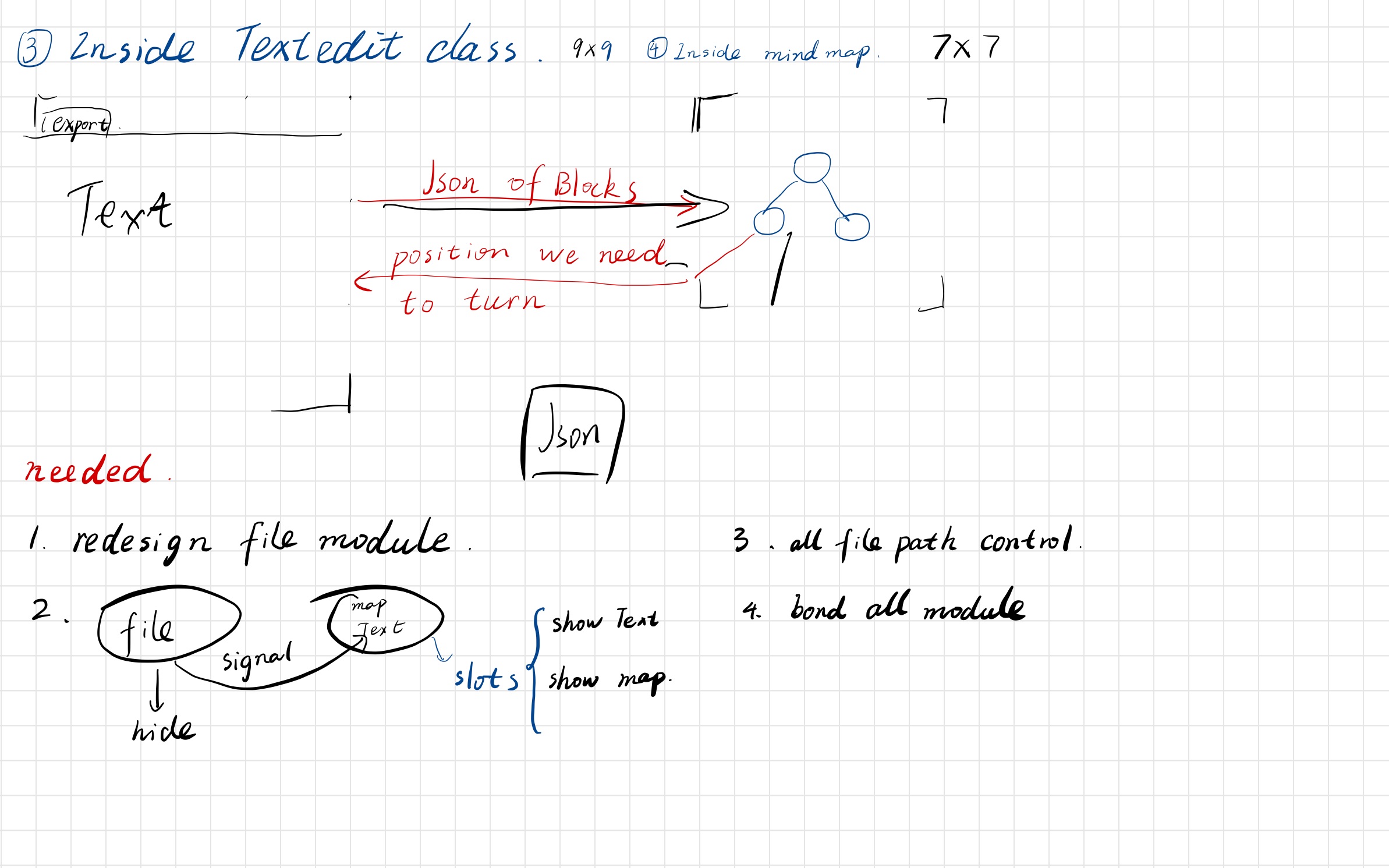
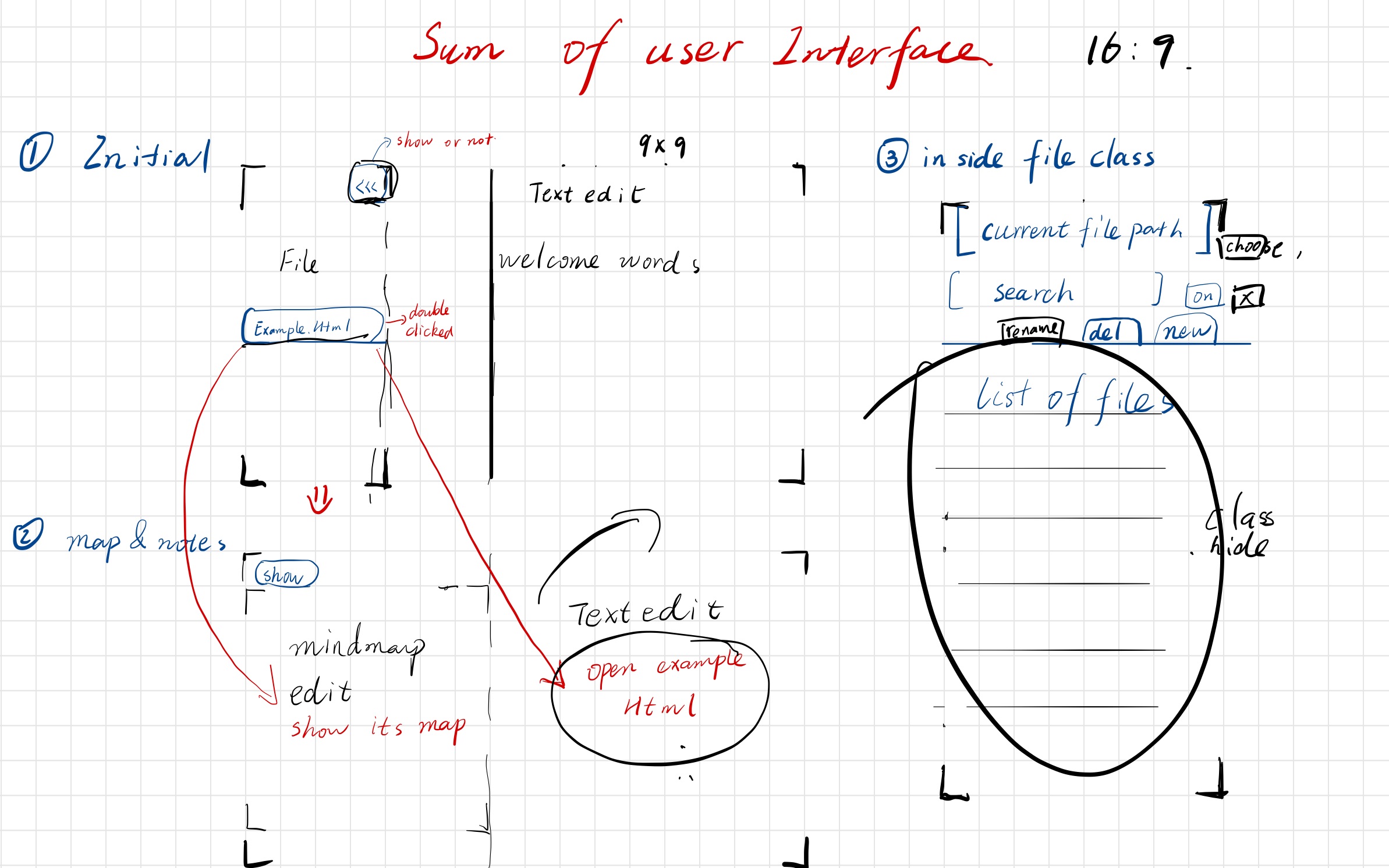
2.根据导图数量的动态导图集合显示

3.导图的编辑功能.

}

}

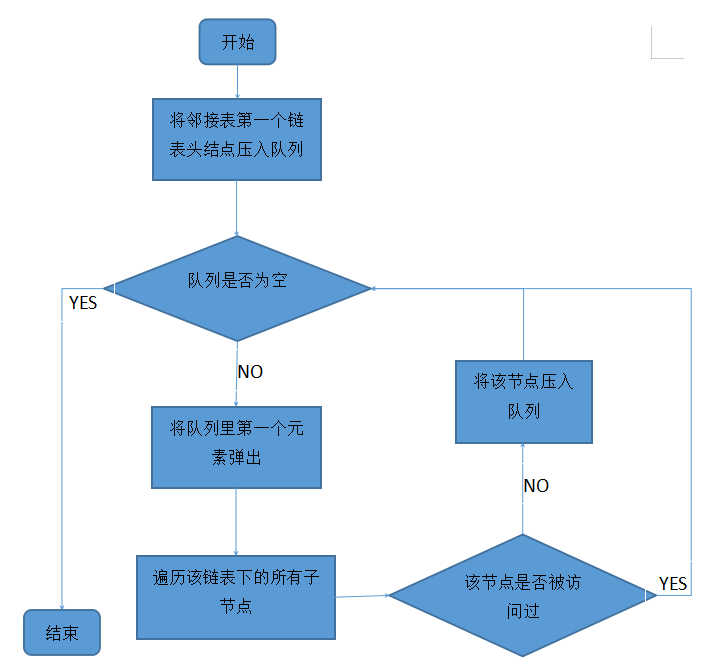
功能模块设计原稿截图:

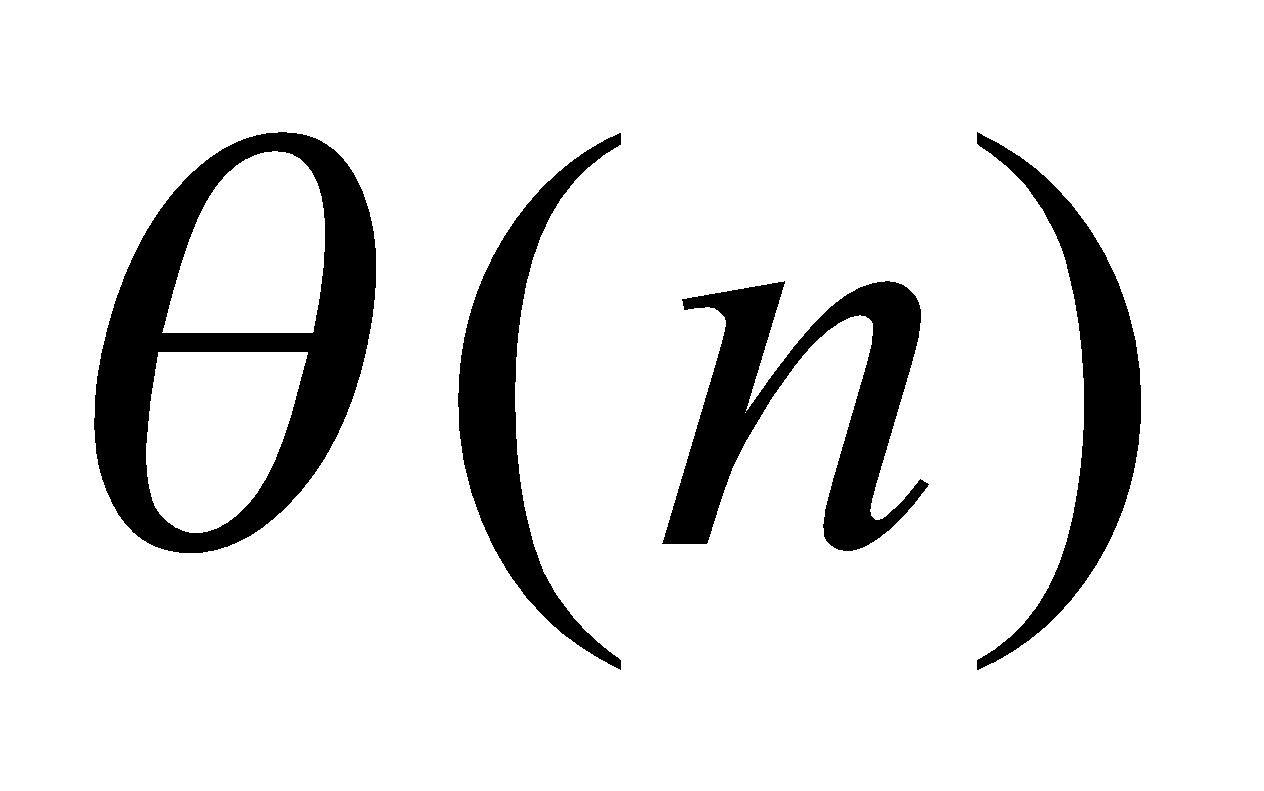
核心算法设计与分析

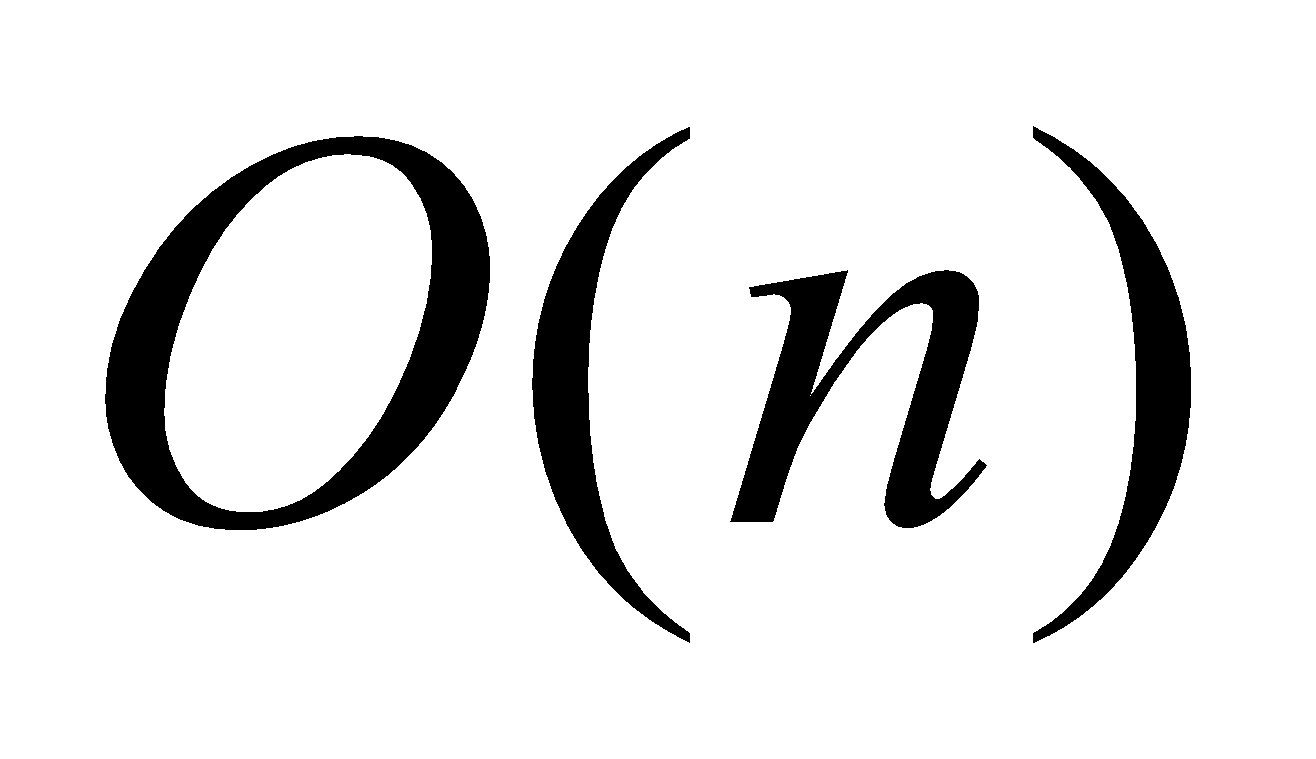
（根据软件功能设计，概述所用到的主要数据结构；用伪代码或程序流程图的形式来详细描述核心算法的功能及过程，并定性分析其时间、空间复杂度）

主要数据结构：栈、队列、数组、树的邻接表、并查集

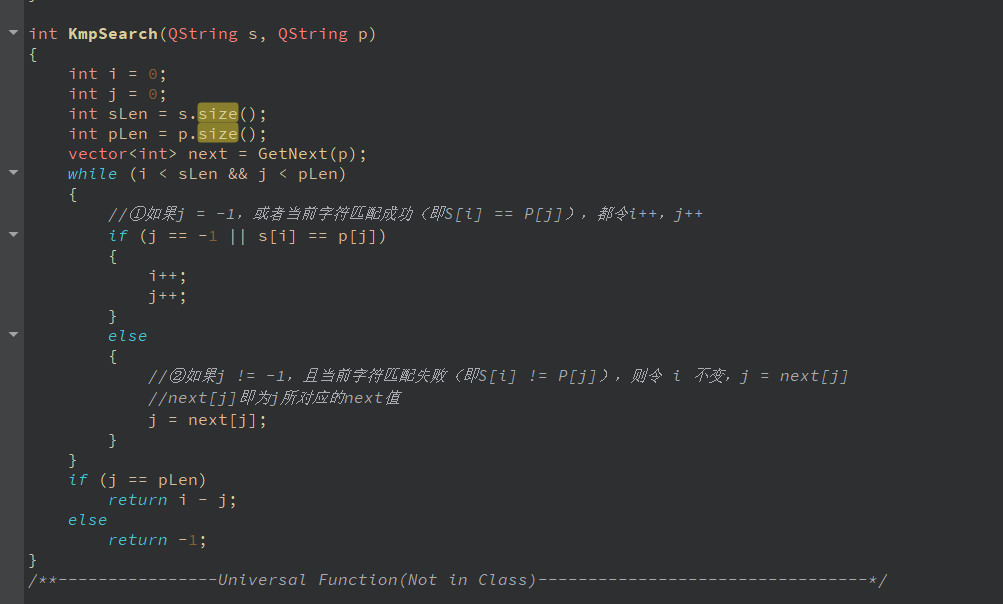
**一、**遍历图的算法：广度优先搜索(BFS)，利用广度优先搜索算法遍历邻接表在涉及到删除节点后所有节点的更新以及将节点信息写入到本地时会用到该算法



时间复杂度：设节点数为n，因为该算法是遍历一遍所有的节点，所以时间复杂度为

空间复杂度：该算法需要一个队列，最坏情况下需要n个，所以空间复杂度为

二、搜索文件名的算法：KMP算法，实现对文件的关键词匹配查找



NEXT-FUNC(P, next)

1.let next[1..m] = -1

2.next[1]=0, next[2]=1

3.for i=3 to m

4. j = i

5. while j != 0

5. if P[j] == P[next[j]]

6. next[i]=next[j]+1

7. break ;

8. else

9. j = next[j]

10. if j == 0 then next[j] = 1

KMP-CMP(T, P)

1. NEXT-FUNC(P, next), j = 0;

2.for T[i] form T[0] to T[n]

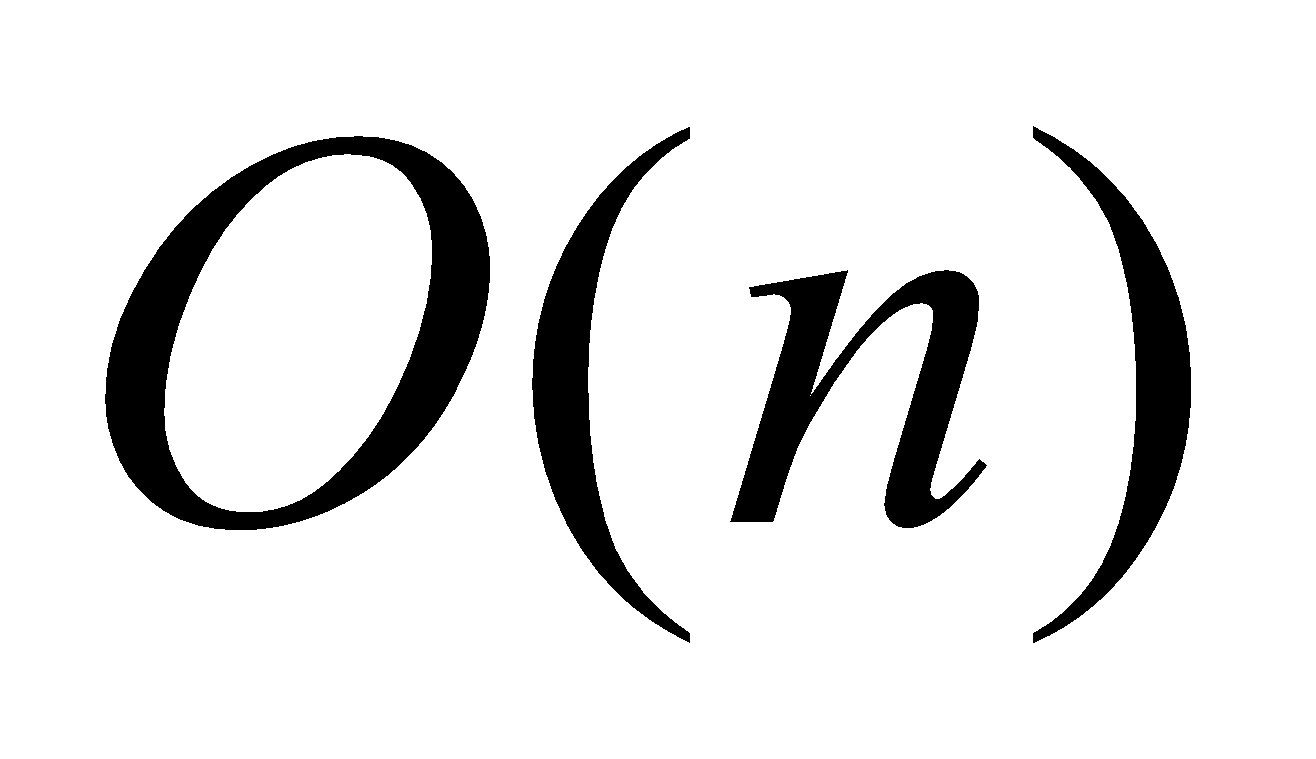
3. if T[i] == S[j]

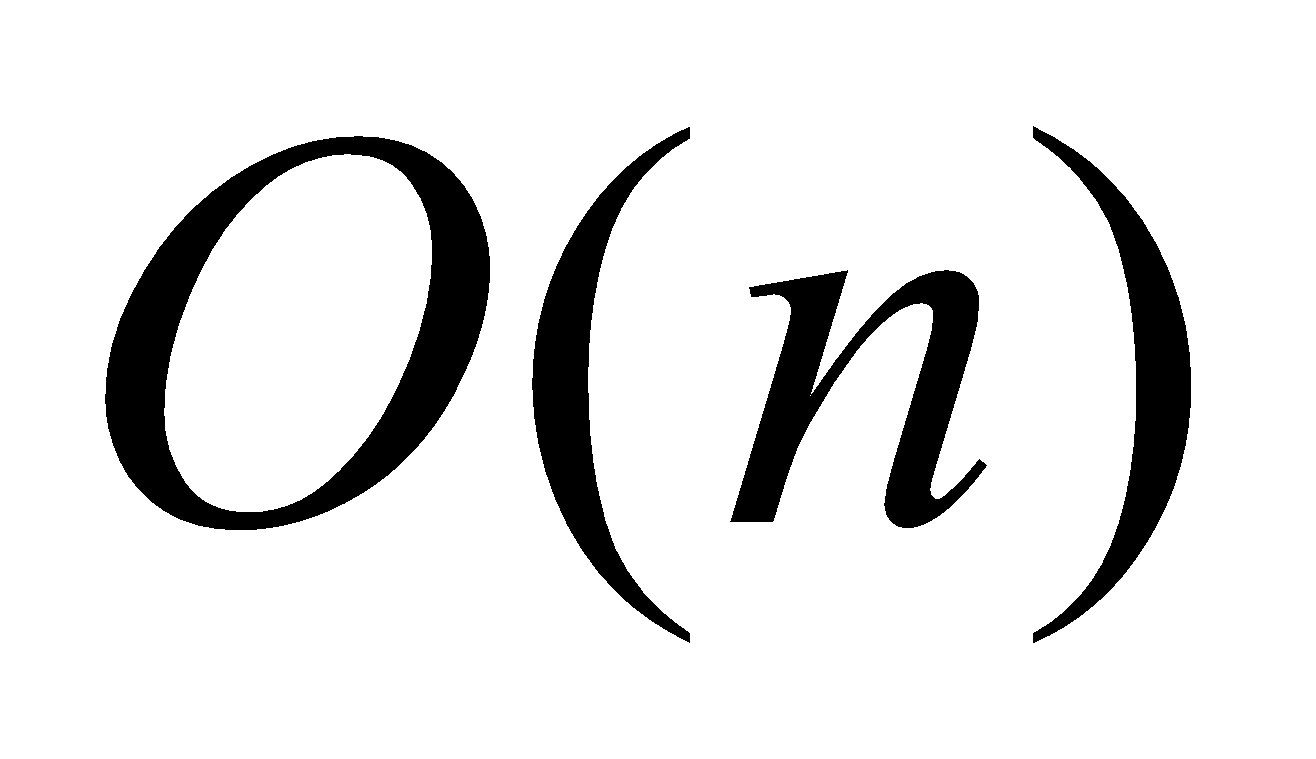
4. if j == m then retrun true

5. then i++ j++

6. else j = next[j];

7. if j == 0 then i++ j++

时间复杂度：设字符串长为n，所以时间复杂度为

空间复杂度：该算法需要一个与被匹配字符串等长的next数组，所以空间复杂度为

系统核心模块实现

（给出编程语言、开发环境及支撑软件、软件系统架构；给出主要数据结构的定义代码以及核心算法对应的关键函数定义代码，包括输入参数的含义、函数输出结果等；核心函数的实现代码段及注解等；主要功能界面截图及对应文字概述等）

开发环境:

在x86 windows 10 上,使用qt creator 4.13(community) 进行开发.

在qt中,选择 c++ 进行逻辑代码编辑,使用qml进行样式编辑.

使用msvc 2019 进行编译.

成品是运行在桌面上的本地exe文件.

核心数据结构代码:

1. kmp

int KmpSearch(QString s, QString p)

{

int i = 0;

int j = 0;

int sLen = s.size();

int pLen = p.size();

vector<int> next = GetNext(p);

while (i < sLen && j < pLen)

{

//①如果j = -1，或者当前字符匹配成功（即S[i] == P[j]），都令i++，j++

if (j == -1 || s[i] == p[j])

{

i++;

j++;

}

else

{

//②如果j != -1，且当前字符匹配失败（即S[i] != P[j]），则令 i 不变，j = next[j]

//next[j]即为j所对应的next值

j = next[j];

}

}

if (j == pLen)

return i - j;

else

return -1;

}

2. 思维导图的保存

void MindMapDialog::onSave( ){

QString str;

QFile f(mapPath);

f.*open*(QIODevice::*WriteOnly* | QIODevice::*Text* ); *//以只写的方式*

QTextStream txtOutput(&f);

*for*(int i=0;i<TreeNode.size();i++){ *//* *num* *x* *y* *father* *position* *string*

str = QString::number(TreeNode[i]->num,10)+" "+QString::number(TreeNode[i]->locatex,10)+" "+QString::number(TreeNode[i]->locatey,10)+" "+QString::number(TreeNode[i]->father,10)+" "+QString::number(TreeNode[i]->position,10)+" "+TreeNode[i]->CurrentText;

txtOutput << str <<"\n";

}

f.*close*();

}

3. 思维导图的读取

void MindMapDialog::onShow(){

QString context;

QFile f(mapPath);

qDebug()<<mapPath +"map path begin ";

f.*open*(QIODevice::*ReadOnly* | QIODevice::*Text*);

QTextStream txtInput(&f);

*while*(!txtInput.atEnd())

{

context = txtInput.readLine();

QStringList strlist = context.split(" ");

int num = strlist[0].toInt();

int lx = strlist[1].toInt();

int ly = strlist[2].toInt();

int father = strlist[3].toInt();

int position = strlist[4].toInt();

QString content;

*if*(strlist.size() == 6) content = strlist[5];

//遍历读取节点信息

*if*(father == num){

Node \*node = *new* Node(*this*);

node->setPos(lx,ly);

node->setBasePoint(node->pos());

m\_scene->addItem(node);

node->num=num;

node->locatex = lx;

node->locatey = ly;

node->father = father;

node->position = position;

node->CurrentText =content;

node->setText(content);

node->setToolTip(QString::number(num,10));

TreeNode.push\_back(node);

AdjacencyTable AT;

Adjacencytable.push\_back(AT);

TreeNumber++;

TreeNodeNumber++;

}*else*{

Node \*fromNode = TreeNode[father];

Node \*node = *new* Node(*this*);

*if* (fromNode) {

QList<Link \*> linksTo = fromNode->getLinksTo();

*if* (linksTo.isEmpty()) {

node->setPos(fromNode->x() + 140, fromNode->y());

node->setBasePoint(node->pos());

m\_scene->addItem(node);

node->setToolTip(QString::number(num,10));

node->num=num;

node->locatex = -1;

node->locatey = -1;

node->father = father;

node->position = position;

node->CurrentText =content;

node->setText(content);

TreeNode.push\_back(node);

AdjacencyTable AT;

Adjacencytable.push\_back(AT);

Adjacencytable[father].son.push\_back(num);

TreeNodeNumber++;

Link \*newLink = *new* Link(fromNode, node);

m\_scene->addItem(newLink);

}

*else* {

Link \*link = linksTo.last();

*if* (link) {

Node \*toNode = link->getToNode();

*if* (toNode) {

node->setPos(toNode->x(), toNode->y() - 100);

node->setBasePoint(node->pos());

m\_scene->addItem(node);

node->setToolTip(QString::number(num,10));

node->num=num;

node->locatex = -1;

node->locatey = -1;

node->father = father;

node->position = position;

node->CurrentText =content;

node->setText(content);

TreeNode.push\_back(node);

AdjacencyTable AT;

Adjacencytable.push\_back(AT);

Adjacencytable[father].son.push\_back(num);

TreeNodeNumber++;

Link \*newLink = *new* Link(fromNode, node);

m\_scene->addItem(newLink);

*//调整布局*

adjustNodes(fromNode, +50);

}

}

}

}

}

}

4. json文件读写

//写入json信息

void MainInfo::on\_pushButton\_clicked()

{

qDebug()<<"maininfo begin";

Info = ui->lineEdit->text();

qDebug()<<arr;

qDebug()<<jsonFile;

qDebug()<<position;

QJsonObject Ablock;

Ablock.insert("position",position);

Ablock.insert("mainInfo",Info);

arr.append(Ablock);

qDebug() << "after";

qDebug()<<arr;

QJsonDocument rootDoc;

rootDoc.setArray(arr);

QByteArray rootStr = rootDoc.toJson(QJsonDocument::*Compact*);

QString jsonString(rootStr);

qDebug() << jsonString;

QFile file(jsonFile);

*if* (!file.*open*(QFile::*WriteOnly* | QFile::*Text*)) {

QMessageBox::warning(*this*, "Warning", "Cannot save file: " + file.errorString());

*return*;

}

*//* *setWindowTitle("save"+fileName);*

QTextStream out(&file);

out << jsonString;

file.*close*();

}

..写入

ui->setupUi(*this*);

path = jsonPath;

QFile file(path);

*if* (!file.*open*(QIODevice::*ReadOnly* | QFile::*Text*)) {

QMessageBox::warning(*this*, "Warning", "Cannot open file: ") ;

*return*;

}

QTextStream in(&file);

QString text = in.readAll();

*//qDebug()<<* *text;*

QJsonDocument doc1 = QJsonDocument::fromJson(text.toUtf8());

array = doc1.array();

qDebug()<<"json begin ";

qDebug()<<array;

*//qDebug()<<jsonFile;*

*//qDebug()<<position;*

ui->listWidget->clear();

*for*(QJsonArray::iterator it = array.begin() ; it != array.end() ; it++){

QString ss = (\*(\*it).toObject().constFind("mainInfo")).toString();

int num = (\*(\*it).toObject().constFind("position")).toInt();

qDebug()<<num<<"\n";

QListWidgetItem \*item = *new* QListWidgetItem;

item ->setText(ss);

item ->setTextAlignment(Qt::*AlignLeft*);

item ->setToolTip(QString::number(num,10));

ui->listWidget->addItem(item);

}

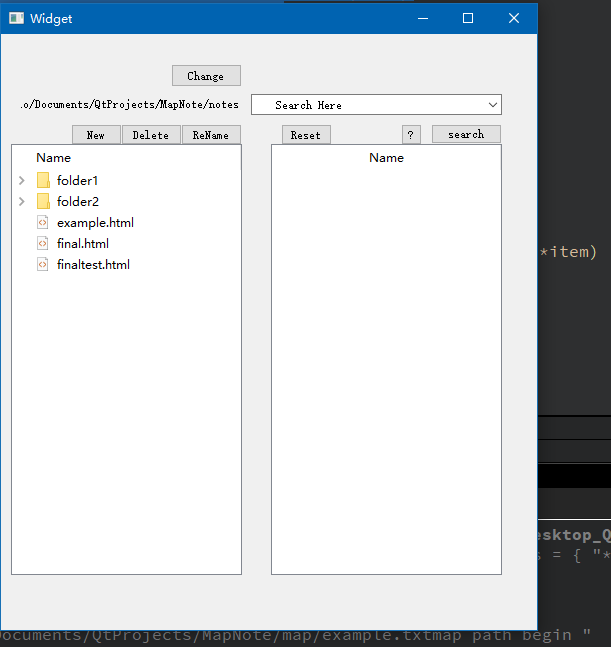
*//qDebug()<<* *"this* *is* *ahhhhhhh";*

file.*close*();

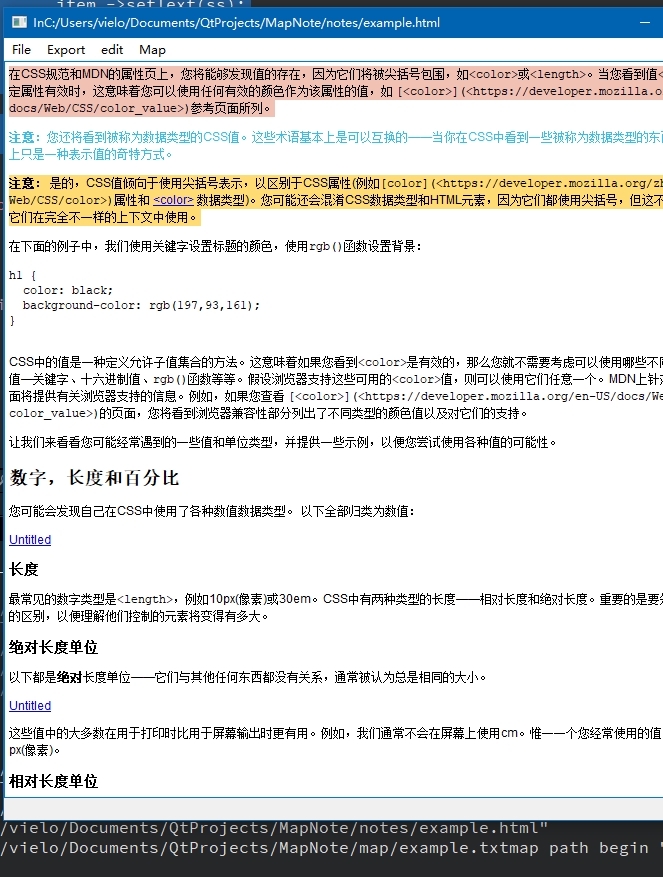
}

模块功能:

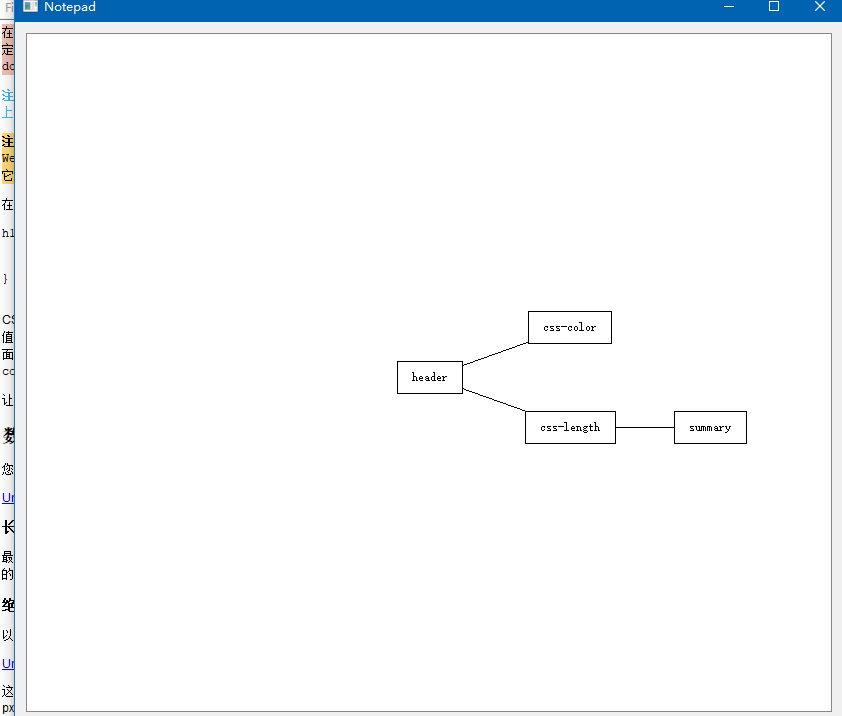
文件管理模块



文本编辑模块



思维导图模块



调试分析记录

（软件开发调试过程中遇到的问题及解决过程；核心算法的运行时间和所需内存空间的量化测定；符合实际情况的数据测试，算法及功能的改进设想等）

鉴于我们组的版本管理做得不是很好,无法追溯到以前版本,所以一下之作分析.

我个人是负责文本编辑模块的编写,主要有一下几个问题:

1. qt 中signal & slot 的使用,刚开始在上手时,对 其不太了解,所以在进行多个窗口交互时,往往没有反应
2. 文本块位置的标记,我一开始的想法是,对每一个html tag 加一个id 属性,从而通过id 确定文本块位置,但随着项目深入,发现我采用的qtextedit 类不支持对tag 自定义属性,如果想要实现该功能,则要更换到webengine 架构以获得更为全面的富文本编辑.最后,通过qtextedit 自带的cusor posotion ,返回当前光标位置达到获得文本块位置的目的,但又几个缺点:
   1. 位置是通过该光标所指字符的位置,而不是该块的位置.
   2. 无法自动更新位置.

运行结果与分析

（要有多组正确的实际测试数据，并给出相应运行结果截图，并对多组结果进行比较分析）

教师指导建议及解决记录

（概述任课教师在开题指导、中期检查和软件验收环节中对课程任务的指导建议及各自所采取的相应解决措施和效果等）

8.1 开题指导及中期检查

8.2 软件验收

1. 总结（收获与体会）

（如实撰写课程任务完成过程的收获和体会以及遇到问题的思考、程序调试能力的培养提升等相关内容；要求不少于500字，严禁雷同）

关于软a实验结束之后的总结,大概有一下几方面体悟.

1. 团队之间的交流与分享.

基本上,我们还是以线下聊天为最基本的交流方式,通过qq进行在线交流,使用github进行代码托管.问题在于大家对git的使用不熟练,并没起到理想之中upload merge 的效果,基本只有单方面的发布,其次,再文件分享方面,缺少一个基于web端的灵活的快速分享平台,导致我们许多资料通过邮件发送.最后,是交流的不充分性,往往我描述的需求和他人所理解的不是同一个东西.

1. 新知识的学习

我们整个平台基于qt和c++ 进行开发,这两方面对我来说都是新鲜领域,实话实说,再整个过程中,c++ 对我而言就是 “c” + “ class” .并没有引用到c++的高级特性.在学习qt时.我的基本流程是先在中文内容中寻找该方面的快速入门文章,之后再通过qt的官方文档进行细致的代码分析.

1. 开发文档的完备性

虽然我认为我在代码工作之前的文档已经比较详细了,但还是有两个大问题.一是在写文档时,我对我们使用的技术根本不了解,对一部分功能的可行性没有验证,导致在后期部分功能实现时遇到较大困难.第二是文档忘记对数据的具体格式和接口进行早起的规划,导致在进行模块组合时十分低效.

1. 进度和版本管理

说来惭愧,我个人的拖延症十分严重,直到国庆在着手进行实际的开发,前期基本没有对组内进度进行追踪,但好在最后大家都即使完成了.版本管理方面,我们的git就是一个摆设,最后还是通过邮件进行组合管理.

1. api 接口的统一和测试

数据交换的格式我们确定下来的时间还是太晚了,导致后续测试的时间不足.其次,老实说,在结题答辩之前,我们并没有进行严谨的测试,仅仅能保证各个功能可以运行,但并没有考虑用户的所有操作,导致程序健壮性不强,这也体现个人在写代码时逻辑的局限性.