# 程序文档

## A组

### UTF-8字符串解析

按照UTF-8编码解析字符串，并封装为Utf8String类型的对象

#### Utf8String

##### 成员类型

|  |  |
| --- | --- |
| 成员类型 | 定义 |
| data\_type | std::u32string |
| value\_type | data\_type::value\_type |
| raw\_type | std::string |
| size\_type | data\_type::size\_type |
| reference | value\_type& |
| const\_reference | const value\_type& |
| pointer | value\_type\* |
| const\_pointer | const value\_type\* |

##### 成员函数

|  |  |
| --- | --- |
| 函数名 | 功能 |
| （构造函数） | 构造Utf8String |
| （析构函数） | 销毁字符串 |
| front | 访问首字符 |
| back | 访问最后的字符 |
| clear | 清除内容 |
| push\_back | 后附字符到结尾 |
| substr | 返回子串 |
| size | 返回字符数 |
| length | 返回字符数 |
| empty | 检查字符串是否为空 |
| raw | 返回std::string类型的版本 |
| c\_str | 返回不可修改的C字符数组版本 |
| find | 于字符串中查找字符(串) |
| operator= | 为字符串赋值 |
| operator[] | 访问指定字符 |
| operator+= | 后附字符(串)到结尾 |
| operator+ | 连接两个字符串或者一个字符串和一个字符 |
| operator== | 以字典序比较两个字符串是否相等 |
| operator!= | 以字典序比较两个字符串是否不相等 |
| operator< | 以字典序比较两个字符串 |
| operator<< | 执行字符串的流输入 |
| operator>> | 执行字符串的流输出 |

### 中文分词

依赖Utf8String类型,对中文基于信息量实现分词操作,并封装成一个可以直接获取分词结果的接口

#### InfoQuantity

封装词语信息量的查询操作

##### 接口

|  |  |
| --- | --- |
| 接口定义 | 功能 |
| double get\_infoquantity(const Utf8String &word); | 接受一个词,并返回该词的信息量 |
| bool count(const Utf8String &word); | 接受一个词,并返回该词在词频文件是否存在 |

#### Segmentation

##### 接口

|  |  |
| --- | --- |
| 接口定义 | 功能 |
| std::vector<Utf8String> segment(const Utf8String &sentence); | 接口接受一个UTF-8编码的Utf8String类型的字符串,并返回分词之后的词语集合.该接口保证返回词语集合的顺序与其在原句中的次序相同. |

### 倒排索引

封装倒排索引,实现如下功能:

1. 倒排索引的建立
2. 根据关键词进行检索
3. 对象到文件的序列化
4. 文件到对象的反序列化

#### FileInfo

描述单个关键词与单个文件之间联系的结构体

包含以下信息:

1. 关键词所在文章的文件路径
2. 关键词在该文章中出现的次数和频率
3. 关键词是否出现在文章的标题中

#### KeywordInfo

描述单个关键词在单个文件中的信息的结构体

包含以下信息:

1. 关键词在该文章中出现的次数和频率
2. 关键词是否出现在文章的标题中

#### InvertedIndex

##### 成员类型

|  |  |
| --- | --- |
| 成员类型 | 定义 |
| key\_type | Utf8String |
| value\_type | std::vector<FileInfo> |
| data\_type | std::map<key\_type, value\_type> |

##### 成员函数

|  |  |
| --- | --- |
| 函数名 | 功能 |
| （构造函数） | 建立倒排索引 |
| ready | 返回该倒排索引是否已建立成功 |
| serialize | 由对象序列化到文件 |
| unserialize | 由文件反序列化到对象 |
| get\_filepaths | 检索给定关键词,返回检索结果,只包含结果的文件路径 |
| add\_files | 根据给定文件夹路径,添加新的索引 |
| add\_file | 根据给定文件路径,添加新的索引 |

### 检索服务器

基于Socket的UNIX域实现进程间通信, 并通过多线程优化多进程访问时的效率

Request

规定与其它进程通信的细节

包含以下信息:

1. 检索的类型
2. 检索的关键词列表

#### SearchServer

封装服务器的行为,只提供监听和启动操作,隐藏实现细节

##### 成员函数

|  |  |
| --- | --- |
| 函数名 | 功能 |
| （构造函数） | 构造SearchServer, 初始化套接字 |
| listen | 开始监听外部请求 |
| run | 进入主循环,处理外部检索请求 |

## B组

### 前端部分

前端部分的文件及其调用关系、输入输出、功能如下表所示

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 文件名称 | 调用关系 | 输入 | 输出 | 功能 |
| index.php | ->SearchEnter.php | 搜索类别、公式、关键词 | GET方式传参到SearchEnter  .php，页面跳转 | 整个搜索引擎的入口 |
| SearchEnter.  php | ->Index2ResultList.php/Index2ConceptMap.php | 搜索类别、公式、关键词 | GET方式传参到Index2ResultList.php或Index2ConceptMap.php，页面跳转 | 1. 根据搜索类别来决定调用结果列表模块还是知识图谱 2. 更改加号不引起GET错误 |
| Index2ResultList.php | GetResultList.php->  LaTexTransfer.php-> | 搜索类别、公式、关键词 | 文件路径、文件名、摘要文本 | 搜索结果列表的生成和显示 |
| Index2ConceptMap.php | GetResultList.php->  LaTexTransfer.php-> | 公式、关键词 | 概念图谱、文件路径、文件名、摘要文本 | 概念图谱的生成和显示 |
| GetResultList.php | EncapAndDecap.php->  UnixDomainSocket.php-> | 搜索类别、公式、关键词 | 包含了文件路径、文件名、摘要文本列表的类 | 封装请求参数和响应数据，中间层 |
| EncapAndDecap | MessageClasses | 搜索类别、公式、关键词/响应数据 | 搜索类别、公式、关键词/响应数据切分数组 | 传输数据和应用数据双向封装 |
| UnixDomainSocket.php | / | 搜索类别、公式、关键词 | 响应数据 | Unix域本地套接字系列操作封装函数 |
| MessageClasses | / | / | / | 定义了包含了文件路径、文件名、摘要文本列表的类 |
| LaTexTransfer.php | / | LaTex公式原始字符串、关键字 | 加上高亮标记的关键字、映射成文件的LaTex公式 | 关键词高亮和LaTex显示映射 |
| Page.php | / | 文件路径 | 文件内容 | 文件具体内容显示页面 |

### 后端部分

#### 摘要求解

##### 模块介绍

AbstractBuilder(); 构造函数

~AbstractBuilder(); 析构函数

void InitKeyword(LISTSTR); 初始化关键词

void ParseFile(LISTSTR); 解析文章

string FileReader(LISTSTR); 读取文章

LISTSTR SentenceFilter(string); 按关键词逐个进行语句切分

PRIQUEUESTR DivideSentence(string, char); 语句切分

float ScoreSentence(string); 语句打分

void CheckSentence(LISTSTR); 避免语句重复

void GetAbstract(LISTSTR); 拼接摘要

string Abstract(); 获取摘要

备注：typedef list<string> LISTSTR;

typedef priority\_queue<STC> PRIQUEUESTR;

typedef list<string>::iterator LISTSTRITER;

##### 功能介绍

1.将一篇文章按照分隔符分成若干个句子

2.统计句子里面关键字出现的次数，和关键字总长度占句子总长度的比值

3.按下列公式计算句子的优先级



4.句子按优先级进入优先队列（排序规则重载为降序），选择topN拼接成为摘要

#### 变量替换

##### 模块介绍

VarIndep(); 构造函数

~VarIndep(); 析构函数

char GetSymbol(string); 返回变量替代符

##### 功能介绍

根据变量出现的次序来依次将它们替换为不同符号（有规律）

#### 运算符优先级

##### 模块介绍

void InitConfigure(); 序列化优先级配置文件

char GetPriority(string, string); 运算符获得优先级

void CheckTable(); 检查配置文件工具

##### 功能介绍

输入LaTex格式的运算符，获取它的优先级。

优先级一览表如下：

|  |  |
| --- | --- |
| 优先级 | 符号列表 |
| 0 | \because \therefore \perp \parallel \neq = \cong > < \leq \geq \le \geq |
| 2 | + - |
| 3 | \* / |
| 4 | \frac \sqrt ^ \sin \cos \tan |
| 5 | ( ) [ ] { } \lvert \rvert |
| 6 | \triangle \square \angle \% |

**2.4 公式转为后缀**

**2.4.1 模块介绍**

MidtoPost::MidtoPost( ) 将操作符与其对应的优先级存于priority map中。

int MidtoPost::compare(stack<string>,string,int) 比较当前操作符与栈顶操作符的优先级大小

list<string> MidtoPost::turn(string) 将Latex公式转为后缀.

* + 1. **功能介绍**

1. 创建3个栈，一个符号栈用来存各种运算符，一个数字栈用来存各种常量，变量，一个表达式栈用来存后缀表达式。遍历字符串，对变量和运算符进行判断，如a到z，A到Z，0到9的认为是变量，其余认为是运算符，对于latex公式中的运算符都会以”\”开头。
2. 对一些特殊字符的判断，如一些歧义符号”+”既可以是中缀运算符又可以是前缀运算符，”-“也是。对\vert绝对值符号，使用时在子公式前后加上此符号，如\vert(a+b)\vert，类似与括号，将他认为是前缀运算符，转为后缀为 a b + \vert。
3. 每一轮遍历比较当前符号与栈顶符号的优先级，每个符号的优先级存在一个priority文件里面，事先会将文件中的符号与其对应的优先级值存到一个map结构中。方便比较。
4. 对于比较结果，若当前符号优先级大于栈顶符号优先级，则将当前符号压入栈。若小于等于，则开始向表达式栈中压（遇到数字直接压），对于括号，中括号，大括号，以及\vert做特殊处理。

**2.5公式解析树**

* + 1. **模块介绍**

void parse\_expr(list<string> , Priority \* )；生成公式树

void set\_pri(Priority \* , base\_node \* )；变量替换

void set\_ord(Priority \* , base\_node \*, VarIndep \*)；公式标准化

**2.5.2 功能介绍**

（1） 遍历二叉树，将变量替换为标识符，并将每个子节点排序

操作符：按算数优先级

变量：按替换顺序

操作符>变量

（2）得到所有子树

**2.6 原始文件的分类整理**

**2.6.1 模块介绍**

map<string,int> mapconcept 存概念及其出现次数。

Typedef struct {}AMGraph 邻接矩阵，存各个概念的关系矩阵

Typedef struct {}relat[MVNum] 用于D3概念图谱生成的数据结构

int find(string,string,int) 查找相关概念在矩阵中的位置

void classifi( ) 将原始文件分为5类，并提取概念关系

* + 1. **功能介绍**

1. 程序运行将遍历file目录夹下的所有文件，对每个文件内容进行遍历
2. 对文件内容进行分析，根据不同的标签，将文件内容分为5类，章节类，概念类，性质类，例题类，练习类。
3. 对概念进行分析，对邻接矩阵进行扩展，最后对用于D3概念知识图谱的内容进行扩展
4. 将分类好的文件按照章节存于source文件夹下。