第29组_概要设计报告

小组成员：

吴承泽　PB19051183

刘伟　　PB19051195

梁峻滔　PB19051175

一、解析过程

在分析markdown文件时，主要采用逐行解析的逻辑，对文本的每一行都进行分析并构建语法树，根据当前行的语法将创建的结点插入到语法树中，最终得到包含markdown文件全部内容的语法树。最后，通过展开语法树，将解析之后的文档用html的形式展示。

二、类的定义

以下类的定义为针对实现对markdown文本的语法解析和到html文本的转换而预想设定，实现时根据实际情况可能会有所不同。

```
class MDtoHTML {
public:
    MDtoHTML(string &filename): _root(Node(NULL)), _filename(filename); //构造函数
    const char* RemoveSpace(char* str); //去除行首的空格，方便之后语法分析
    bool IsCutLine(char *str);          //判断是否是水平分割线
    void Insert(Node* Node, char *str); //逐字符插入节点，str为当前行内容
    void DFS(Node* root);               //深度优先遍历root，并将语法树转换成HTML
    void Trans(); //对传入的md文件进行逐行语法分析，构造语法树，展开后生成正文部分的html源代码
    void Derivehtml(); //生成html文件的head和end部分，将head、正文、end连接、导出html文件
    void Destroy(Node *root);           //销毁语法树
    ~MDtoHTML();
private:
    Node* _root;       //语法树根节点
    string _filename; //文件名
    string _content;  //存放HTML文档内容
}
```

三、主程序流程

```
int main() {
    MDtoHTML md("myfile.md"); //将要转换的markdown文件作为参数，实例化
    md.Trans(); //生成正文部分的html源代码
    md.Derivehtml();  //生成目标html文件
    return 0;
}
```

四、模块调用关系

`Trans()`是主要的转换函数，对要转换的markdown文件进行语法分析，期间需要调用`RemoveSpace()`、`IsCutline()`、`Insert()`等成员函数来构造语法树，之后需要调用`DFS()`来展开语法树生成正文部分的html源代码，之后调用`Derivehtml()`加上head和end部分后导出一份完整的目标html文件。