实验说明

161250194

张贝贝

目录

[0前言 2](#_Toc26589)

[1输入说明 2](#_Toc2476)

[1.1内部嵌入 2](#_Toc18972)

[1.2输入内容 2](#_Toc29582)

[1.3修改说明 2](#_Toc40)

[2输出说明 3](#_Toc9214)

[2.1输出内容 3](#_Toc28222)

[2.2输出截图 3](#_Toc30249)

[3核心算法思想说明 4](#_Toc21138)

[3.1生成First集 4](#_Toc10311)

[3.2生成Follow集 4](#_Toc20493)

[3.3生成Parsing Table 5](#_Toc24830)

[3.4语法分析过程 5](#_Toc19598)

[4数据结构 5](#_Toc18722)

[5收获与思考 5](#_Toc1610)

[5.1收获 5](#_Toc9241)

[5.2思考 5](#_Toc16644)

[6联系方式 5](#_Toc25769)

0前言

·本次实验选取了最后一种方案，即构造自己的YACC：遵从LL(1)分析法，手动编程实现了从语法规则获得First集，Follow集，从而生成Parsing Table并最终得到test语句（一个简单的例子）的语法分析过程并打印。

·语法规则较为简单，暂不需要进行词法分析，单纯展示LL(1)分析法的全部过程。

1输入说明

1.1内部嵌入

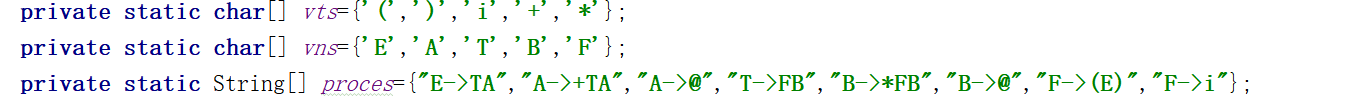
·考虑到构造自己的YACC这一整套流程要体现的东西较为繁杂，故选择了简单的语法规则，简洁明了，易于观察过程。

·语法规则已写在代码头部，作为内部数据，不需要外部输入。

·产生式已进行过预处理，消除了左递归。

1.2输入内容

·语法规则



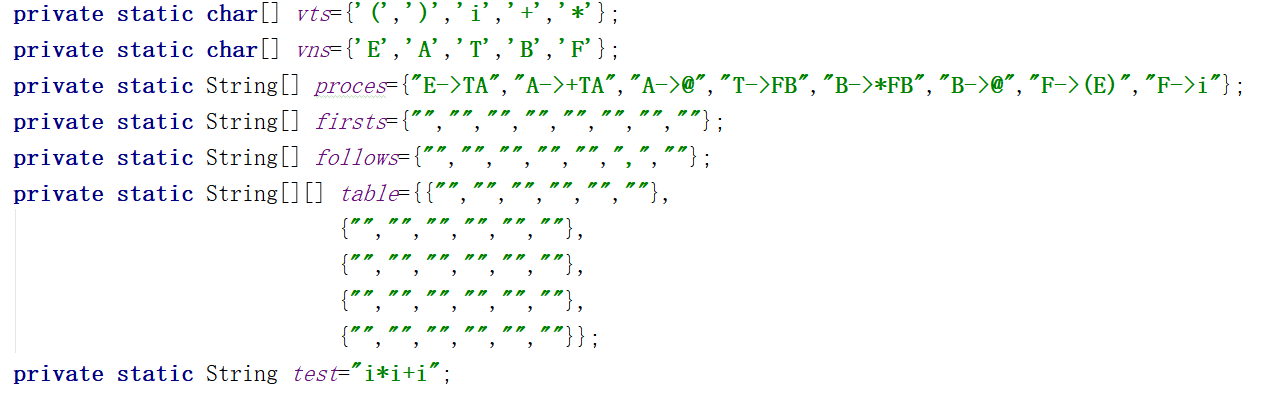
·测试用例

test

1.3修改说明

·因为是内嵌的语法规则，已经作为常量，所以修改起来较为麻烦。

·如需修改规则进行测试，请将一下部分都进行统一（包括更新测试用例）：



注：代码逻辑部分不需要担心，没有使用常量

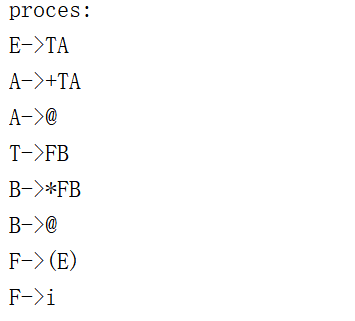
2输出说明

2.1输出内容

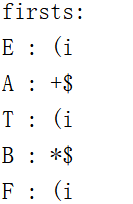
·为了体现LL(1)分析法的全过程，方便观察，将在控制台输出产生式，First集，Follow集，Parsing Table以及test语句语法分析出入栈过程。

2.2输出截图

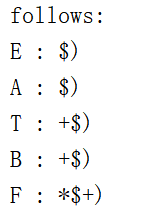
·产生式



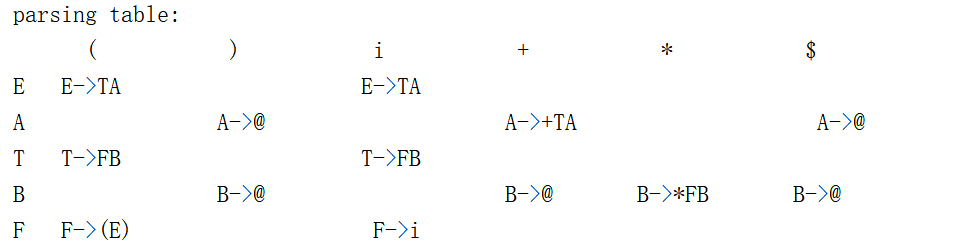
·First集



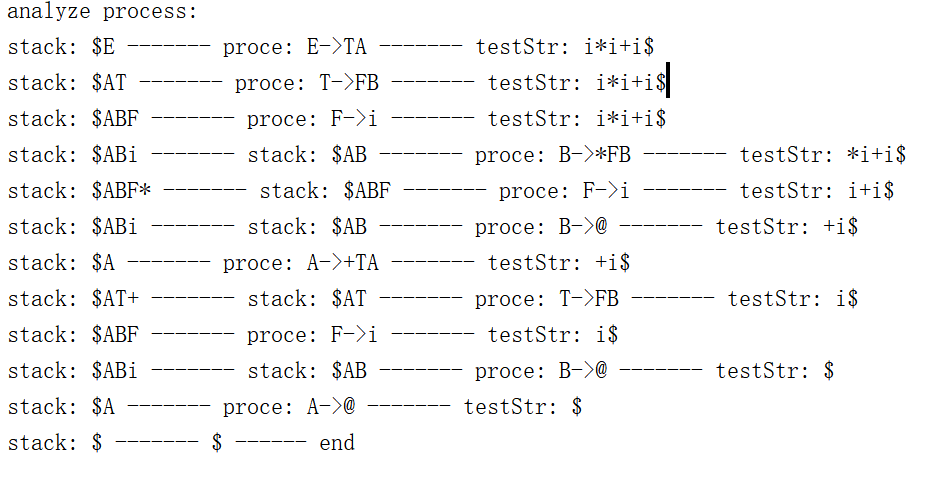
·Follow集



·Parsing Table



·语法分析过程



3核心算法思想说明

3.1生成First集

·遍历每一个产生式

·如果产生式右边第一个字符为终结符，则将其并入左部非终结符的first集合中

·如果产生式右边第一个字符不是终结符：

·求该非终结符的first集合（没有$），并将其并入左部非终结符的first集合中

·如果该非终结符存在空产生式，就循环并下一个字符的first集合（终结符为它自身）

·如果该非终结符不存在空产生式，则跳出循环

·如果循环到产生式结尾，即产生式右部都可以为空，则将$加入左部非终结符的first

集合中

3.2生成Follow集

·对于整个文法的开始符S，将$并入S的follow集合中

·遍历每一个产生式，观察右部的每一个非终结符后面的第一个字符：

·如果是终结符，则将其并入该非终结符的follow集中

·如果是非终结符，获得其first集（去除$）并入当前非终结符的follow集中

·如果后面的非终结符存在空产生式，则循环观察下一个字符

·如果不存在空产生式，则跳出循环

·如果循环到产生式结尾，即后面的部分都可以推出空，则该非终结符的follow集合需

要并左部非终结符的follow集

·注：设置follow集的方法执行了两遍，因为可能存在A的follow集合依赖B的follow集合但B还没有求Follow集合。因为是用并集的方法进行添加follow，不会出现问题。

3.3生成Parsing Table

·遍历产生式

·如果是空产生式，获得左部的Follow集合，放入表中对应的位置

·如果不是空产生式：

·如果右部第一个字符为终结符，将产生式直接放入表中对应位置

·如果右部第一个字符为非终结符，获得其First集，将产生式放入表中对应位置

·如果右部非终结符的First集中存在$，则循环获取下一个字符的First集，将产生式放

入表中对应位置

·如果右部非终结符的First集中不存在$，跳出循环

3.4语法分析过程

·先将$与开始符压栈，并在test语句后加$

·对当前test中的目标字符与当前栈顶（并出栈）进行对比，如果相同则目标右移，否则查询Parsing Table获得对应产生式并逆序压栈

·如果上两种情况皆不存在，则报错，是不符合语法的语句

·循环第二步直到栈空

4数据结构

·没有生成语法树，所以不需要构造复杂数据结构

5收获与思考

5.1收获

·自己构造的语法分析器虽然非常简易，但完全加深了我对LL(1)文法的理解。从产生式生成First集合、Follow集合，然后获得Parsing Table，最后生成语法分析过程，让我从更加底层的角度认识了编译器，我相信这对我以后的编程活动会大有助益。

5.2思考

·在编造这个语法分析器时，我一直在纠结是否需要进行交互（外部输入），后来选择内部规定语法规则，一方面是觉得这个作业的重点是分析过程，而不是分析对象，只要过程正确的体现出来就是达到了作业的目的。另一方面，这毕竟只是一个简易的语法分析器，为了减少突发状况，方便检查（比如助教如果构造了某某语言的文法让这个分析器分析，可能会出bug，因为具体语言分析还得先做对应的词法分析），最后还是选择了定义好语法规则。

6联系方式

如果出现任何问题，麻烦助教大大联系我

QQ：1292155474

邮箱：[1292155474@qq.com](mailto:1292155474@qq.com)

电话：18018691650